

REPLICATION IN BANDWIDTH-SYMMETRIC BITTORRENT NETWORKS

M. Meulpolder, D.H.J. Epema, H.J. Sips

Parallel and Distributed Systems Group
Department of Computer Science, Delft University of Technology, the Netherlands
{M.Meulpolder,D.H.J.Epema,H.J.Sips}@tudelft.nl

ABSTRACT

The popular and well-known BitTorrent peer-to-peer protocol offers fast file distribution in a highly scalable way. Several studies have investigated the properties of this protocol, mostly focusing on heterogeneous end-user environments such as the Internet, with asymmetric connections. In this paper however, we focus on the usage of the BitTorrent protocol in homogeneous local environments with symmetric bandwidth properties. Compared with a traditional client-server setup, the use of BitTorrent in such settings can offer huge benefits in performance and scalability, allowing bandwidth sharing and high speed file distribution. We aim to improve the performance of such networks with a novel mechanism for replication using so-called *replicators*, which replicate a subset of the files in the system. A mathematical model of the resulting *Replicated BitTorrent* is presented and validated by emulation. Furthermore, we present simulation results that provide insight in the performance of Replicated BitTorrent networks with dynamic peer arrivals and departures. The results show that Replicated BitTorrent significantly improves download times in local bandwidth-symmetric BitTorrent networks.

1. INTRODUCTION

Over the last few years, the BitTorrent [1] protocol has become widely used and has grown into a topic of increasing interest in the research community. Among the present day peer-to-peer (P2P) protocols, BitTorrent is clearly the most popular, especially for file sharing over the Internet. The most important reasons for its popularity are its outstanding scalability characteristics, as well as its *tit-for-tat* mechanism [2] which ensures that while downloading a file, users are automatically sharing parts of the file with others. Recently, various papers have been published [3, 4, 5] which contain theoretical models, simulations, and measurements considering the BitTorrent protocol, its performance, and its usage patterns.

978-1-4244-1694-3/08/\$25.00 ©2008 IEEE

Up until now, however, the focus regarding the application of BitTorrent and the research into its properties have been centered around file sharing by heterogeneous end-users on the Internet. A very important consideration in such environments is the often asymmetric nature of Internet connections such as ADSL. Since the BitTorrent protocol is based on the mutual exchange of file pieces by current downloaders, as well as on a certain amount of altruism by finished downloaders, the asymmetry in download and upload speeds of end-user connections is an important assumption in the analysis of the performance and scalability of the protocol. Furthermore, the behavior of peers in a ‘general’ file sharing context is very hard to predict.

This paper is based on a different kind of environment for the application of BitTorrent: local environments with symmetric end-user connections. Important examples are corporate and academic LANs, in which P2P techniques have hardly been applied so far. Yet, in such environments BitTorrent could prove to be the ideal protocol for rapid local file distribution in a highly scalable way. Due to the BitTorrent protocol, users who are downloading the same file at the same time gain a huge profit from each other. As presented in [6], the resulting download times are almost constant with respect to the number of downloaders. Even though there are no tailored implementations of BitTorrent for this context yet, many interesting possibilities can be imagined, such as high-quality multimedia services which use BitTorrent in the background for the actual content distribution.

In our work, we consider local environments in which a collection of files is available for all users in the network. In the case of a *push* model, where a centralized authority decides at a specific point in time to distribute specific files, network-level multicasting techniques could be applied. However, we focus on a *pull* model, where users can decide when to download and what to download out of the collection. A practical example is a university campus, where educational materials

and multimedia files have to be available for all students. Currently, the common approach is to make all files available at a (collection of) central server(s), from where they can be downloaded using the FTP or HTTP protocol. Instead, we consider the use of the BitTorrent protocol for such transfers, in such a way that both the server(s) and the end-user machines transparently act as peers in the resulting peer-to-peer network. As our main contribution, we present, analyze, and simulate a novel replication mechanism which highly improves the performance and scalability of such bandwidth-symmetric BitTorrent networks: *Replicated BitTorrent*. We introduce the concept of *replicators*, which are special peers that replicate a subset of the files in the system, thereby increasing system performance. A model is devised in order to analyze the influence of these replicators, and is validated by emulation using real BitTorrent clients. Furthermore, we provide insight in the performance of Replicated BitTorrent in networks with dynamic arrivals and departures of peers. We have simulated Replicated BitTorrent networks with both homogeneous and Zipf-distributed file popularities. We have assessed various network configurations in order to compare client-server, BitTorrent, and Replicated BitTorrent performance. The results show that replicators significantly increase the performance, leading to download speeds that are many times higher than those in a regular BitTorrent system.

2. REPLICATED BITTORRENT

In this section, we present our replication mechanism for bandwidth-symmetric BitTorrent networks. It has to be noted that throughout the rest of this paper, we make use of standard BitTorrent [1] terminology that can be found in many other publications on the subject. We will briefly introduce the relevant concepts of BitTorrent to those not yet familiar with the subject: in BitTorrent a file is divided into *chunks*, which are then *bartered* between peers that are interested in the file; a *leecher* is a peer who is in the process of downloading a file but has not yet finished; a *seeder* is a peer in possession of the whole file and sharing it with the network; a *swarm* associated with a file is the collection of peers leeching or seeding that file. Peers discover other peers in the network by requesting IP addresses from the *tracker*, the only central component in a BitTorrent network. The tracker keeps track of the seeders and leechers of each file.

As mentioned in the introduction, we consider local BitTorrent environments in which a collection of files has to be available for all users in the network. Some of the peers, the *sources*, provide a central point for injecting content and ensure that at least one copy of every file is

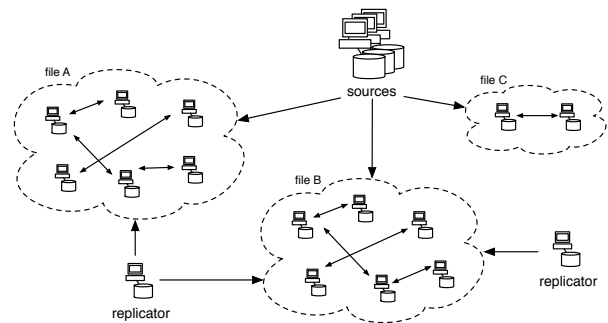


Figure 1: Example of a simple Replicated BitTorrent system with a few sources, three active swarms associated with three different files, and two replicators which seed various files.

available. We now distinguish three types of seeders: (1) the sources, that seed all files and are assumed to be always online, (2) the peers that have finished a download and remain online for a certain amount of time, and (3) *replicators*. A *replicator* is a peer that is added to the network, and that is seeding a subset of the files that are available in the system. In Figure 1, an example of a simple Replicated BitTorrent system with replicators is displayed.

At the BitTorrent level of a particular file swarm, a replicator is not different from a general BitTorrent seeder. However, we focus on the system as a whole and seek to influence the *number* of seeds of particular files, in such a way that the resulting download speeds in the system are as high as possible. The mechanism of replicators is designed to influence *which files* are seeded by *which number of seeders* based on the file popularity in the system, without changing the BitTorrent protocol itself. In a small network, replicators can be used to reduce the load of the network as a whole. In a larger network, consisting of different segments with possibly different file popularity characteristics, replicators can be placed in the different segments to specifically reduce the load of the each segment in an optimal way. The contents of the replicators depend on the popularity of files in the system or network segment. In our current implementation, replicators are notified of the file requests in the system by the tracker, which we extended to provide this functionality (the overhead as compared to regular tracker functionality is very small). Based on these file requests replicators monitor the file popularity and decide which files to seed according to their *replication policy*. In this paper, we evaluate different file popularity distributions, request arrival rates and replication policies.

3. ANALYSIS

In this section we present a model to analyze the performance of Replicated BitTorrent in a given system with known swarm properties. We validate the model by emulation using real BitTorrent clients.

3.1. Model

We consider a Replicated BitTorrent system with a fixed number of sources and a fixed number of replicators. Each of the sources seeds all of the files available in the system. Each of the replicators seeds a subset of the available files. In the model, different replicators can seed different subsets of files, though the number of files seeded is constant over all replicators. In Section 3.4, we discuss the restricted case in which all of the replicators seed the same files. We introduce the following notation:

- \mathcal{F} The set of files seeded by the sources.
- F The size of \mathcal{F} .
- S The number of sources.
- N_f The number of peers downloading file f .
- M_f The number of finished peers still seeding f .
- R The number of replicators.
- Q The number of files seeded by each replicator ($Q \leq F$).
- c_s The upload bandwidth of the sources.
- c_r The upload bandwidth of the replicators.
- c The upload/download bandwidth of regular peers.
- β_f The fraction of replicators seeding file f .

We assume that a peer is downloading *at most* one file at a time so that its full download bandwidth is available for each single download. Furthermore, we do not take bartering overhead into account, that is, we assume that the bandwidth used for bartering fully results in effective download bandwidth. Since we assume that each source seeds the same files, we can without loss of generality assume that there is only one source ($S = 1$).

3.2. Scalability of the number of leechers

We will show that in the case of bandwidth symmetry, the effective download bandwidth of a leecher is independent of the number of leechers in a swarm.

Theorem 1 *Let in a bandwidth-symmetric BitTorrent swarm of a file f the total available bandwidth of all seeders be u . Suppose that the leechers of this file are not seeding or leeching any other files. Then the fol-*

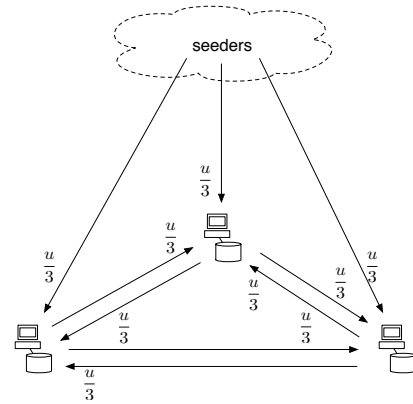


Figure 2: Illustration of the proof of Theorem 1 for a swarm with 3 leechers.

lowing holds for the download speed d of any leecher: $d = \min\{u, c\}$.

Theorem 1 makes sense intuitively, since all bandwidth that is received by a leecher can be made available to other leechers. The proof is stated below. An illustration of the proof is displayed in Figure 2.

Proof of Theorem 1

Suppose that the number of leechers in a particular swarm is n , and that every leecher obtains a download speed of u/n from the seeders. We assume first that $u \leq c$. Since there are only $(n - 1)$ leechers to barter with, not more than a bandwidth of $(n - 1)(u/n)$ can be used effectively for bartering with the other leechers. The maximum download speed then becomes

$$d = \frac{u}{n} + (n - 1)\frac{u}{n} = u.$$

When we do not take bartering overhead into account, and when all peers have at least one chunk to share, we can assume this maximum is reached. It follows that when a total speed of c is obtained from the seeders, the download speed of a leecher is maximal. Hence, $u > c$ will always lead to an effective download speed of c . \square

3.3. Analysis of the download speed

In current BitTorrent implementations, the upload bandwidth of a peer is evenly shared among all files that it seeds or leeches. Since this also holds for the sources, each source provides a bandwidth of c_s/F to each swarm in the system. With $S = 1$, it follows from Theorem 1 that any leecher of any file obtains an effective download speed from the source of $d = \min\{c_s/F, c\}$.

The total available amount of bandwidth to a swarm of a specific file f consists of three contributions. First

of all, the swarm receives its share of bandwidth from the source, which is c_s/F . Secondly, there are M_f peers that have finished downloading and are now seeding file f . Finally, there is the bandwidth contribution from the replicators. For a file f , there are $\beta_f R$ replicators that are seeding f , each of which divides its bandwidth over the Q files that it is seeding.

Because of Theorem 1, each of the leechers in the swarm will obtain an effective download speed that equals the total available bandwidth of all seeders. Hence, the effective download speed d_f of a leecher in the swarm of file f is

$$d_f = \min\left\{\frac{c_s}{F} + M_f c + \beta_f R \frac{c_r}{Q}, c\right\}. \quad (1)$$

3.3.1. Optimal number of replicators

Since the download bandwidth of the downloading peers is bounded by c , the increase in bandwidth due to the replicators is maximized when

$$\beta_f R \geq \left((1 - M_f)c - \frac{c_s}{F}\right) \frac{Q}{c_r}. \quad (2)$$

If the number of replicators is such that given a certain β_f , the resulting download bandwidth for file f is maximized, we call the replicator configuration *optimal* for f . If the configuration is optimal for all f , we call the configuration *optimal*. We define the *optimal number of replicators* R_{opt} as the minimum integer value of R such that Eq. (2) holds for all f with $\beta_f > 0$.

3.3.2. Speedup of a swarm

We calculate the speedup s_f of a leecher of file f as the fraction $d_f/(d_f|_{R=0})$. According to Eq. (1) this results in

$$s_f = \min\left\{1 + \frac{\beta_f R F c_r}{Q(c_s + F M_f c)}, \frac{F c}{c_s + F M_f c}\right\}. \quad (3)$$

3.3.3. Speedup of the system

If we want to consider the system as a whole, we have to take into account all leechers in all swarms. The *average speedup* \hat{s} over all leechers in the system is then

$$\hat{s} = \frac{\sum_{f \in \mathcal{F}} N_f s_f}{\sum_{f \in \mathcal{F}} N_f}. \quad (4)$$

3.4. Special case

In order to provide more insight into the practical implications of the model, we consider the special case in

which all of the following assumptions hold: (1) all peers in the system have the same speeds ($c_s = c_r = c$); (2) finished downloaders immediately leave the system, which implies that $M_f = 0$ for all f ; (3) the replicator configuration is *homogeneous*, i.e., all replicators seed the same files. A file f is therefore seeded by all or by none of the replicators. Hence, for every f it holds that either $\beta_f = 1$ or $\beta_f = 0$.

Let N be the total number of downloaders for all files. For the homogeneous configuration, we define the *replication fraction* $\alpha \equiv Q/F$ as the fraction of the available files that is seeded by the replicators. Eq. (4) is now reduced to

$$\hat{s} = 1 + \frac{1}{N} \sum_{\{f: \beta_f=1\}} N_f \min\left\{\frac{R}{\alpha}, F - 1\right\} \quad (5)$$

According to Eq. (2), the speedup in this case is maximal if $R \geq R_{opt} = \lceil \alpha(F - 1) \rceil$.

3.5. Model validation

In the previous sections we have analyzed the download speed in a single swarm of file f , and the speedup of the system achieved with the replicators. In order to validate the model, we have performed emulations of real swarms using real BitTorrent clients. We have built a swarm emulator for this purpose, which is based on the simulator used in [7]. The emulator creates an artificial BitTorrent swarm and initiates a BitTorrent download.

We have emulated swarms with various numbers of replicators. In every emulation there is a single source ($S = 1$) that seeds 8 files ($F = 8$), and all speeds are homogeneous ($c = c_r = c_s = 5$ MB/s). Finished peers do not remain in the system ($M_f = 0$ for all f). We use only one leecher in every emulation, since the download speed in a swarm is independent of the number of leechers. The emulation results for different swarms are combined to determine the system speedup for $\alpha = 1/8, 1/4, 1/2$ and 1, each with a varying number of replicators. The resulting values are compared with the values predicted by Eq. (5).

In Figure 3, the system speedup is plotted, as well as the speed of leechers that reside in a swarm with replicators. The corresponding values along with the theoretical predictions are displayed in Table 1. It can be observed in the table that the values for the speedup resulting from the emulations are very close to the values predicted by the model. It is reasonable to expect a small bartering overhead, as well as an error margin in the emulations. Note that when $R \geq R_{opt}$, the emulation values remain practically constant, as predicted by the model.

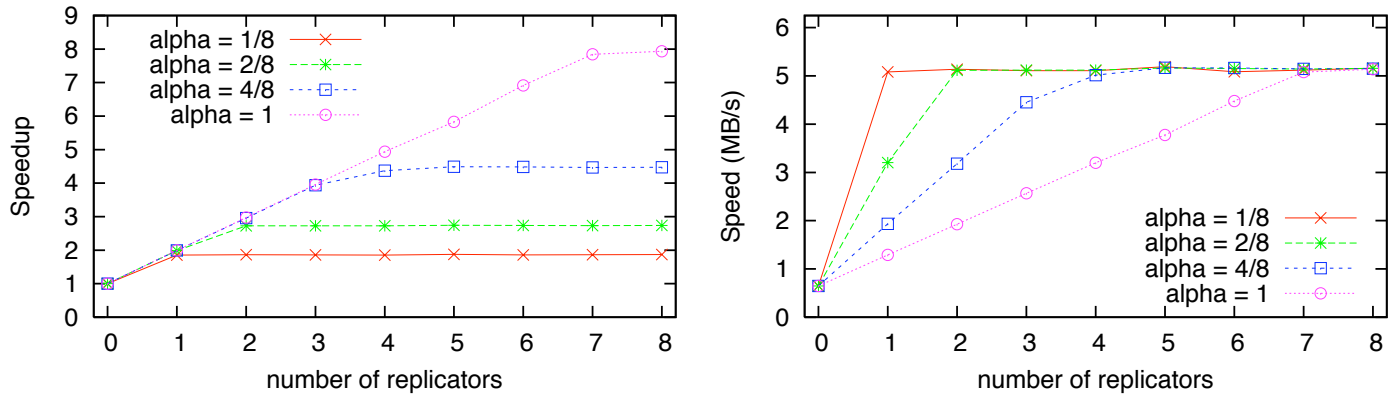


Figure 3: The system speedup (left) and the download speed of leechers in a swarm with replicators (right) for various values of the replication fraction α .

α	1/8		1/4		1/2		1	
	sim	model	sim	model	sim	model	sim	model
1	1.856	1.875	1.986	2.000	1.994	2.000	1.995	2.000
2	1.866	1.875	2.725	2.750	2.955	3.000	2.974	3.000
3	1.861	1.875	2.725	2.750	3.937	4.000	3.960	4.000
4	1.861	1.875	2.726	2.750	4.369	4.500	4.938	5.000
5	1.876	1.875	2.743	2.750	4.487	4.500	5.829	6.000
6	1.856	1.875	2.738	2.750	4.484	4.500	6.916	7.000
7	1.863	1.875	2.735	2.750	4.470	4.500	7.840	8.000
8	1.870	1.875	2.740	2.750	4.475	4.500	7.932	8.000

Table 1: The system speedup with varying numbers of replicators. The cases where the replicator configuration is (theoretically) optimal are printed in boldface.

4. EVALUATION

The validated model as presented in the previous section offers an accurate way to determine the speeds in a system where the specific swarm properties for every file f are known. We now want to provide insight in the behavior of a realistic system in which swarms and download speeds dynamically change due to arriving and departing peers. We have performed simulations of various configurations of bandwidth-symmetric networks, both with and without replicators. In this section we describe the simulation setup and we present the simulation results.

4.1. Simulation setup

We have simulated a bandwidth-symmetric BitTorrent network with one source ($S = 1$) that has $F = 40$ files available. There are 10 replicators ($R = 10$) that seed a fraction $\alpha = 1/4$ of these files. The replicators are homogeneous: each replicator seeds the same set of files. Given these parameters, the number of replicators is optimal ($R = R_{opt}$). All files have a size of 500 MB and all upload and download bandwidths in the system are

set to 5 MB/s. In the system, peers requesting a file arrive according to a Poisson process with arrival rate λ . We have performed simulations for values of λ between 10 and 480 arrivals/hour. In each of the simulations we have measured the download time of a file as soon as a steady state was reached, i.e., a state in which the available download speed for the file no longer changed over time.

Each incoming request is associated with a specific file according to a given file popularity distribution. We have simulated two different file popularity scenarios: a homogeneous file popularity where every file has an equal probability of being requested, and a Zipf-distributed file popularity. In both cases, the file popularity does not change over time. In the homogeneous case, the replicators seed the same set of 10 arbitrary files. In the Zipf-distributed case, we have evaluated the system with various replication policies. We have assessed five different network configurations: (1) A client-server network (CS); (2) A BitTorrent network without replication where all peers leave immediately after finishing their download (BT); (3) A Replicated BitTorrent network where

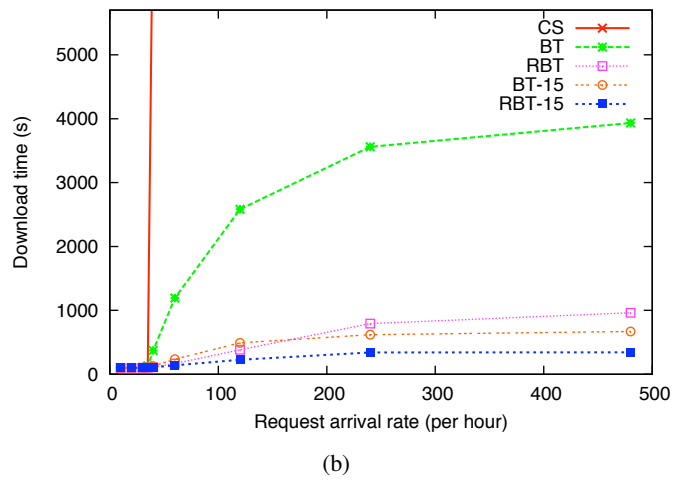
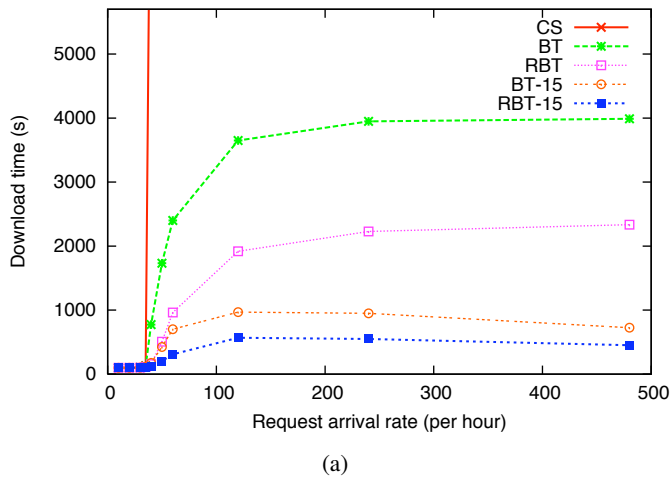


Figure 4: The download times in different bandwidth-symmetric BitTorrent networks with (a) homogeneous file popularity, and (b) Zipf-distributed file popularity.

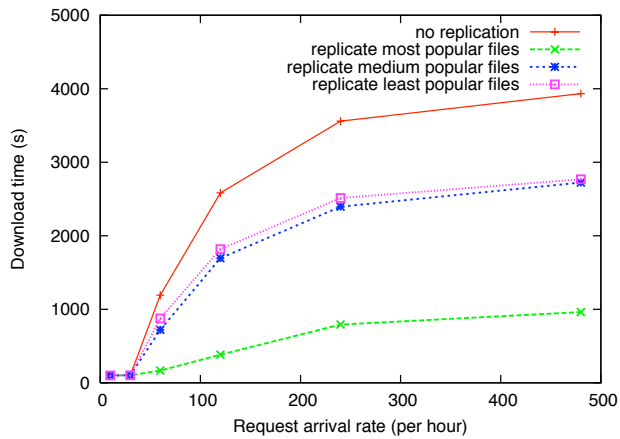


Figure 5: Comparison of different replication policies under a Zipf-distributed popularity distribution.

all peers leave immediately as well (RBT); (4) A BitTorrent network with 15 minutes seeding of each finished peer (BT-15); (5) A Replicated BitTorrent network with 15 minutes seeding of each finished peer (RBT-15).

4.2. Results

In Figure 4, the steady state download times are displayed for the two popularity distributions and the various network configurations. In these results, the replication policy is such that the 10 most popular files are seeded by the replicators. It can be observed that the client-server download time stays low until the arrival rate reaches a certain threshold, after which it explodes. This is to be expected since if requests arrive with a higher rate than the rate with which they can be finished by the server, the download speed drops to zero. The plain BitTorrent con-

figuration shows a far better scaling behavior in which the download time converges to 4000 seconds. This again is intuitive since with a high enough arrival rate, swarms will be active for each of the 40 files continuously and the bandwidth of the source will be divided equally over these swarms. With a resulting bandwidth of $5/40$ MB/s, a download of 500 MB naturally takes 4000 seconds.

For both file popularity distributions, the configurations with replicators perform much better than the other configurations. The most significant speedup is obtained in the Zipf case with immediate leave of finished peers, where replication leads to download times that are close to 25% of those in a network without replication. If finished downloaders are seeding as well, download times are still only 50% of the times without replication. In the homogeneous case, the trends suggest that in a configuration where finished downloaders remain in the system, the effect of replication will eventually diminish. This is intuitive since replicators are seeders, and if the number of other seeders in the system grows, the relative contribution of the replicators becomes less significant. Figure 4 (a) shows a decreasing download time when finished peers remain seeding, since the total contribution in bandwidth of finished peers naturally grows with the arrival rate of new peers.

In Figure 5, the download speeds for three different replication policies are displayed for a Zipf distributed file popularity scenario with no seeding of finished peers: (1) replicating the most popular files; (2) replicating medium popular files; (3) replicating the least popular files. As can be expected, replicating the most popular files yields the best results. The swarms of more popular files are bigger and therefore the speed with which these swarms are served benefits larger numbers of peers.

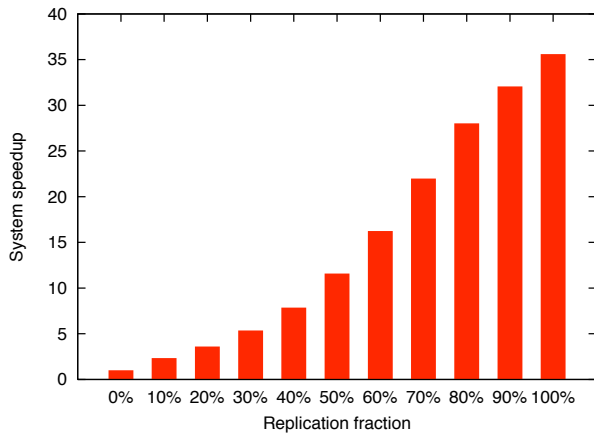


Figure 6: Influence of the replication fraction on the speedup of a system with 10 replicators and a Zipf distributed file popularity with 240 requests per hour. The replicator policy is to seed the most popular files.

This policy performs far better than the others while there seems hardly a difference between replicating a set of medium popular files or the set of least popular files. This can be explained by the nature of the long tailed Zipf distribution where the difference in popularity between the most popular files and medium popular files will be far greater than the difference in popularity between medium popular files and the least popular files.

In Figure 6, the influence of the replication fraction on the system speedup is displayed for a system with 10 replicators, a Zipf distributed file popularity, and 240 request arrivals per hour. A higher replication fraction effectively implies that more swarms obtain added bandwidth, but the bandwidth added per swarm is less. It is important to observe that when the replication fraction gets larger, the 10 replicators cause a speedup that exceeds 11, the value that seems intuitive at first sight. This can be explained by the fact that all regular BitTorrent clients divide their bandwidth equally over all files that they are seeding (instead of over all *peers* that are downloading from them). As a result, the bandwidth available to the most popular swarm is equal to the bandwidth available to the least popular swarm. By adding bandwidth to the more popular swarms, the overall BitTorrent distribution becomes more efficient.

5. DISCUSSION

In our model and experiments, we have taken various assumptions of the network environment and file popularity conditions. We will discuss the most important assumptions and their implications in a more general way here.

5.1. Bandwidth homogeneity

It is assumed throughout the model that all peers have bandwidth symmetry, and that this bandwidth is homogeneous throughout the system. Even though bandwidth symmetry is realistic for most LANs, there might in reality be bandwidth bottlenecks in the network. First of all, different segments of the network might be connected by backbones that form a bottleneck. Secondly, organizations might be spread over multiple buildings or locations, connected via the Internet. In these situations, the scalability of Theorem 1 will be limited by upper bounds depending on the location of the various peers in a swarm. When a specific network structure is known, the model can be adapted to this network by incorporating scalability limitations in the equations. It follows naturally that as long as the upper bounds are not reached by the sources and the downloaders, the download speeds can be increased by using replicators.

5.2. Replicator contents

Since we have analyzed the replicators in a general context, we have assumed homogeneous replicators which all seed the same files. In a real system however, it might be highly advantageous to allow different replicators to replicate different files. Especially when replicators have different locations in the network, replicators can be dedicated to replicate files that are popular in specific network segments. They might even be added and removed automatically depending on the load in the system, providing effective load balancing. Such systems are hard to analyze a-priori without knowing the details of the context, since many complex assumptions would have to be made about the network structure, the popularity of different files within different network segments, and the location of replicators throughout the system. However, when the environment and conditions are known in detail, our general model can first be applied separately to each segment of the network, after which the resulting sub-models can be joined in a more complex, system-wide model.

6. RELATED WORK

As mentioned in the introduction, most papers on BitTorrent focus on heterogeneous end-users sharing files on the Internet. A notable exception is the work presented in [6] and [8], where BitTorrent-based data distribution on LAN-based desktop grids is studied. The authors show by experiments that BitTorrent clearly outperforms FTP for the dissemination of large files over a LAN, and present an enhancement of the protocol that improves the

performance for small file distribution as well. However, replication mechanisms such as the one we present in this paper are not considered.

Replication and caching have been widely researched in a variety of contexts. None of the work however addresses BitTorrent. Research until now has focused on (earlier) P2P protocols that are based on searching an instance of a requested file and directly transferring it from a particular location. In [9], the effect of the number of replicas on search performance in unstructured P2P networks is investigated. This work is continued in [10] and [11], where it is argued that a proportional replication scheme minimizes the download time, the workload, and the used network bandwidth. The authors furthermore show that near-proportional replication can be obtained by using an LRU cache replacement policy. In [12], various replication strategies are compared as well and a *square-root* replication strategy is presented, which is argued to be better than both uniform and proportional replication. In [13] and [14], the effect of various replication strategies on file availability and reliability is studied.

Our work is specifically focused on replication in BitTorrent, which is very different from the P2P protocols analyzed in earlier work. We present a BitTorrent model with explicit replication, and we exploit BitTorrent specific swarm characteristics to optimize the performance and scalability of content distribution in LANs.

7. CONCLUSIONS

In this paper, we have presented a novel mechanism for replication in bandwidth-symmetric BitTorrent networks: *Replicated BitTorrent*. The concept of *replicators* is presented, which are peers that are added to a BitTorrent network, and that automatically replicate a subset of the files that are available in the system, thereby increasing the download speeds of downloaders in the system. We have presented and validated a mathematical model of Replicated BitTorrent which allows us to compute the download speeds and assess the benefit of replicators. Furthermore, we have simulated Replicated BitTorrent in a dynamic setting where download speeds and swarm properties change due to arriving and departing peers. The simulation results clearly demonstrate the significant speedup of Replicated BitTorrent over regular BitTorrent.

References

- [1] BitTorrent. <http://www.bittorrent.org/>.
- [2] B. Cohen. Incentives build robustness in bittorrent. In *Workshop on Economics of Peer-to-Peer Systems*, Berkeley, USA, May 2003.
- [3] D. Qiu and R. Srikant. Modeling and performance analysis of bittorrent-like peer-to-peer networks. In *Proc. of ACM SIGCOMM 2004*, Portland, Oregon, USA, August 2004.
- [4] J.A. Pouwelse, P. Garbacki, D.H.J. Epema, and H.J. Sips. The bittorrent p2p file-sharing system: Measurements and analysis. In *Proc. of the 4th Int'l Workshop on Peer-to-Peer Systems (IPTPS'05)*, February 2005.
- [5] A.R. Bharambe, C. Herley, and V.N. Padmanabhan. Analyzing and improving a bittorrent network's performance mechanisms. In *Proc. of IEEE Infocom 2006*, Barcelona, Spain, April 2006.
- [6] B. Wei, G. Fedak, and F. Cappello. Collaborative data distribution with bittorrent for computational desktop grids. In *Proc. of the The 4th International Symposium on Parallel and Distributed Computing (ISPDC'05)*, pages 250–257, Washington, DC, USA, 2005. IEEE Computer Society.
- [7] P. Garbacki, A. Iosup, D. Epema, and M. van Steen. 2Fast: Collaborative downloads in p2p networks. In *Proc. of the 6th IEEE International Conference on Peer-to-Peer Computing*, Cambridge, September 2006.
- [8] B. Wei, G. Fedak, and F. Cappello. Scheduling independent tasks sharing large data distributed with bittorrent. In *Grid Computing Workshop*, pages 219–226. IEEE Computer Society, 2005.
- [9] S. Tewari and L. Kleinrock. Analysis of search and replication in unstructured peer-to-peer networks. In *Proc. of ACM SIGMETRICS 2005*, 2005.
- [10] S. Tewari and L. Kleinrock. On fairness, optimal download performance and proportional replication in peer-to-peer networks. In *Proc. of IFIP Networking 2005*, pages 709–717, 2005.
- [11] S. Tewari and L. Kleinrock. Proportional replication in peer-to-peer networks. In *Proc. of IEEE INFOCOM 2006*, 2006.
- [12] E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks. In *Proc. SIGCOMM'02*, Pittsburgh, USA, August 2002.
- [13] G. On, J. Schmitt, and R. Steinmetz. The effectiveness of realistic replication strategies on quality of availability for peer-to-peer systems. In *Proc. of the 3rd International Conference on Peer-to-Peer Computing (P2P'03)*, 2003.
- [14] R. Bhagwan, D. Moore, S. Savage, and G. Voelker. Replication strategies for highly available peer-to-peer storage. In *Proc. of FuDiCo: Future directions in Distributed Computing*, 2002.