Load Balancing through Automated Replication in Unstructured P2P File Sharing Systems

Cristina L. Abad[1]

[1]Facultad de Ingeniería en Electricidad y Computación (FIEC)
Escuela Superior Politécnica del Litoral (ESPOL)
Campus Gustavo Galindo, Km 30.5 vía Perimetral
Apartado 09-01-5863. Guayaquil-Ecuador

cabad@fiec.espol.edu.ec

Abstract. *Current peer-to-peer file sharing systems rely on the users to decide which files to replicate. Replication is done implicitly by the users' decision to download and share a file. Due to free-riding, some files end up not being hosted by enough peers to readily satisfy all concurrent download requests. Some peers can become overloaded with download requests that could be satisfied by other peers if they hosted replicas of the requested file(s). It is desirable to have high availability of files, so that every download request can be readily satisfied. Furthermore, it is desirable that files are shared by a sufficient number of "good" peers (nodes with high-speed connection that are likely to stay on-line for a long period). This paper presents a solution for the problem of load balancing in peer-to-peer file sharing systems, based on the automated replication of files into "good" peers to improve average download times and file availability. Simulation results show that this solution is able to reduce the average download time of files and helps balance the load of peers serving files.*

1. Introduction

Peer-to-peer (P2P) alternatives to applications that previously were server based have emerged in the past few years. As the number of peers participating keeps growing, new and more efficient techniques for handling their interaction need to be adopted.

This paper focuses on P2P file sharing applications such as Gnutella. In these applications, peers act as servents (clients and servers at the same time), downloading files from other peers and allowing other peers to download files from their machines. This approach to the file distribution problem has proved to be successful, and the number of peers participating in this activity keeps growing with time [Ripeanu et al. 2002]. To be able to keep up with their fast-growing pace, these systems need distributed and automated solutions to improve their scalability, usability and response times. P2P file sharing applications are frequently built on top of unstructured overlay networks. Other P2P systems as Kazaa have a basic hierarchy notion: the use of super-peers who manage a few number of peers but only with respect to the forwarding of control messages. These are still considered unstructured networks because there is no coupling between topology and data location [Cohen and Shenker 2002, Lv et al. 2002].

While in an ideal world, every user shares the files he or she downloads and stays connected to the P2P network for long periods, in practice these systems present many

problems like having a large number of free riders[1] and low bandwidth or low connectivity peers. Low bandwidth peers, such as modem users, are not able to act as servers and are not able to let other peers download files from them because the large size of current media files would make this operation unacceptably slow. Low connectivity peers are peers that stay connected to the P2P network only for short periods at a time. These peers usually connect to the system to download a few files and then disconnect [Saroiu et al. 2002b]. In [Saroiu et al. 2002a] the authors observed that in P2P systems, less than 20% of the requests result in a successful transaction. Most P2P requests are met with a "service unavailable" response, suggesting that P2P servers are often saturated.

The proposed solution is for each peer to monitor its load and decide to replicate a (popular) file before becoming overwhelmed with download requests. The file is replicated into a random "good" peer. Model analysis and simulation results show that the solution is able to reduce average download times and balance load in peers when the system is overloaded with download requests.

2. Related Work

File replication [Ranganathan and Foster 2001] and load balancing [Byers et al. 2003, Rao et al. 2003, Karger and Ruhl 2004, Bienkowski et al. 2005] have been much studied in structured peer-to-peer systems. While these papers can provide some ideas on how to address these problems in unstructured P2P systems, most of their approaches are not applicable to unstructured peer-to-peer systems.

Previous related work in unstructured P2P systems address several issues related to the ones analyzed in this paper. In [Ranganathan et al. 2002] the authors propose a method to achieve a desired level of data availability through the use of a model with which peers can decide when and where to replicate files. Thus, replication is done pro-actively and not re-actively. Cohen et al. [Cohen and Shenker 2002] try to minimize maximum (or average) search/query size, instead of balancing load or reducing average download time, by replicating pointers (indexes) instead of copies (file replicas). "Query size" is defined in as the number of peers that need to be visited before finding one that hosts the file. Schmitt et al. [On et al. 2003] take an availability-centric view on quality-of-service in which a model and heuristics are used for dynamic placement of replicas. Their focus, limited to file availability, yields a scheme with different characteristics than the one presented in this paper.

Load balancing in [Roussopoulos and Baker 2006, Suri et al. 2004] is only concerned with distributing the download requests among the different peers that already serve a file, but it is not concerned with replicating files in the network to improve availability. In a P2P system that is not being overloaded by download requests this method may reduce average download time of files. But if the download requests are too many and not so many peers are sharing the desired file(s), then, distributing the download requests more fairly will not reduce the average download time of files because the peers hosting the popular file(s) will be too overloaded.

[1] "Peers that free ride on Gnutella are those that only download files for themselves without ever providing files for download by others" [Adar and Huberman 2000]

3. The Solution

The proposed solution for file load balancing in P2P file sharing networks is to have each peer monitor its load and have it replicate its most requested file(s) to other peers before becoming overwhelmed with download requests.

The following are three different approaches to distributed load balancing:

1. Each peer can monitor its serve load and automatically replicate a file when some threshold is exceeded.
2. Each peer can periodically monitor its files and try to achieve an optimal number of replicas for each file it hosts, based on a model of the system.
3. Have an agent explore the network and determine when/where replicas are needed.

Approach 1 is the one followed in this paper and will be better explained later in this section. Approach 2 is taken in [Cohen and Shenker 2002]. The problem with it is that the behavior of the P2P system could change and the defined model is not able to adapt to it. Approach 3 is usually rejected due to the associated security issues. The mobile code of the agent needs to be executed at each peer and viruses could easily be propagated through it.

The design of the file replication scheme can be divided into three parts: when to replicate, what to replicate, and where to replicate. Next, we describe the proposed solution (Approach 1) with respect to each of these three parts.

3.1. When to replicate

Two different criteria for when to replicate could be used: replicate some file(s) when the bandwidth used by the peer exceeds some threshold, or replicate when the number of requests received per interval exceeds some threshold. In the current scheme, for simplicity, a peer decides to replicate a file when the number of requests received per interval exceeds some threshold.

A different approach to the problem of when to replicate is taken in [Ranganathan et al. 2002]: peers constantly check if the currently available number of replicas (in the whole system) of each file they host is appropriate or not. If the number is too low, the peer replicates the file. The authors also suggest a small variant in which file availability is only checked when a file is requested. This approach depends on a model that estimates the appropriate number of replicas of a file in the system; it is unclear if the model is appropriate or not. Another problem is that many peers may decide to replicate the same file at the same time.

3.2. What to replicate

Once a peer decides to replicate, it must decide what to replicate. The proposed approach is to replicate the most popular file that the peer hosts. File popularity can be determined by keeping track of the number of requests for each file in some period. Since the popularity of files is likely to change in time, it is not wise to consider their lifetime popularity but the recent popularity of files.

It is likely that with time some replicas will become less accessed until a point when it would be wise to eliminate them. A replica aging mechanism should be implemented to avoid the problem.

3.3. Where to replicate

A "good" peer is selected among the connected peers. A "good" peer is found as follows. First, from the possible candidates to host a replica (the candidates are those peers that do not host it already) the best ones according to their bandwidth is chosen. From those, the ones with low connectivity are discarded and a random peer is selected from the ones left.

Note that some underlying mechanism must be provided by the P2P system to enable us to find possible hosts for a replica and what their bandwidth is. Another problem with this method is that it assumes that peers are willing to cooperate. While in real life this may not be the case, it is reasonable to assume that there is a large number of peers that are willing to cooperate [Saroiu et al. 2002b, Golle et al. 2001]. Furthermore, the most cooperative peers are the ones with high bandwidth and high connectivity. Section 3.5 further discusses this issue.

Another issue to be considered is that the peer selected to receive the file may have no more free space. There are two ways around this: select another peer, or erase a replica in that peer (the least popular replica could be deleted).

Another measure for "good"-ness suggested in [Wilcox-O'Hearn 2002] is seniorness of peers. According to [Wilcox-O'Hearn 2002], a problem with MojoNation was that it did not discriminate against newly joined nodes. The length of time that a node has been continuously connected to the network is a good predictor of the length of time that it will remain connected in the future.

3.4. Other issues: File updates and file replacement

As in [Ranganathan et al. 2002], this work assumes that the data shared is read-only. Thus, replica updates and consistency issues do not need to be considered. This assumption holds in current P2P file sharing systems in which users share files that do not change with time (music, video, books, etc.). If files change, a version number can be used to differentiate each version.

The scheme presented in this paper could potentially fill-up the storage space of the "good" peers. Some replicas should be deleted to make space for new replicas. The aging of unused replicas can be used to decide which files to eliminate. As new files come, they should replace the least recently used one(s). This assumes temporal locality, which has been proved to hold in the Gnutella network in [Markatos 2002, Zeinalipour-Yazti and Folias 2002].

3.5. Motivation to participate

It has already been stated that many peers are free riders and choose not to serve files for other peers to download. Its is thus logical to ask: Why would a peer participate in a P2P file sharing automated replication system? In other words, if participating in the load balancing solution will mean that other peers will upload popular files into them, why would they be willing to accept them? It is reasonable to believe that a large number of peers will participate in the load balancing solution for the following reasons:

- Ignorance – Most of the users do not change the default parameters of their applications. If the P2P file sharing software default settings allow the automatic replication of files, most users will leave it that way.

- Altruism – Many peers participating in the file sharing system chose to serve files just because they want to improve the overall value of the system [Golle et al. 2001].
- Understanding of the advantages of participating – Aside from the inconvenience of receiving replicas from other peers, participating in the load balancing solution yields several advantages such as the ability to relieve some of the burden of serving popular files and improved download times.
- Discovery – When a peer is overloaded, it replicates its most popular file into another peer. Users that participate in the load balancing solution could browse the replicas they host and learn what is popular in the system. Since replicas are popular files, browsing through them is a good way to discover files the user might be interested in. This is a useful capability by itself, but it would be even more useful if the peers are participating in a community-based P2P file sharing system, where each community shares a common interest, e.g. doctors, artists, or more specific interests like in Usenet. The value of virtual communities in P2P networks is analyzed in [Sakaryan et al. 2004].

Finally, incentives could be provided to encourage participation, like giving preference to those peers on the download queues (e.g., eDonkey [Tutschku 2004]), or even use some kind of stronger enforcement method, like BitTorrent's tit-for-tat mechanism [Pouwelse et al. 2005].

3.6. Additional requirements

The proposed solution to the problem of reducing average download times in P2P file sharing systems is a simple solution that does not impose significant overhead on the peers. From the peer's perspective, the extra space needed for the replicas is the main problem, but if that were an issue, a condition can be easily accommodated to assign bounds to the replica space.

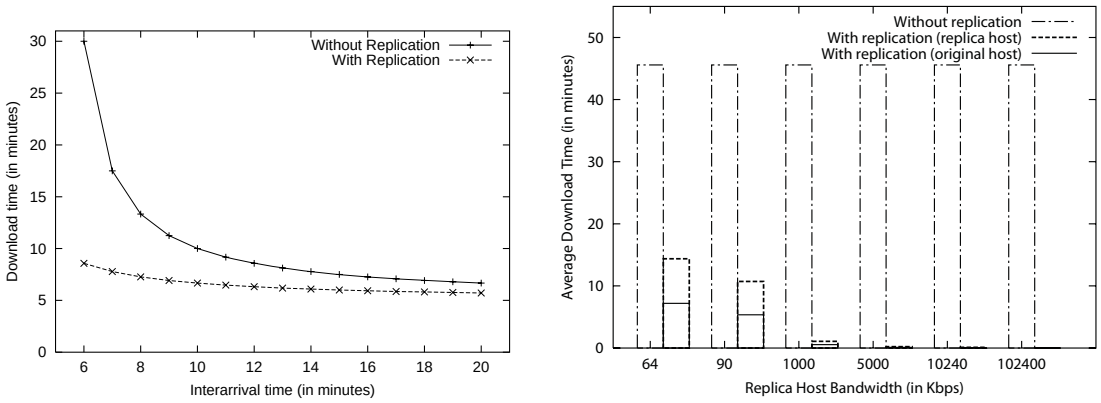
With respect to the underlying P2P network, the proposed solution is independent of the choice of P2P network. It is only required to have a mechanism for a peer to learn about other peers (if they host a file or not, and their bandwidth and connectivity). Given that any P2P file sharing network must have some form of search method implemented, this method can be used with some minor modifications to obtain the desired information.

The main metric for deciding if a peer is "good" or not is its bandwidth. There are two ways to handle this issue. The first is to have the peers report their bandwidth. This method is currently used by many deployed P2P file sharing networks. The problem with this alternative is that unless some incentive is given to the peers for correctly reporting their bandwidth, some tend to explicitly lie. A second alternative is to use some probing technique. This alternative is more accurate but adds complexity and slows down the process.

4. Model

This section presents a queuing theory model that helps understand the impact of using the proposed automated file replication solution.

For simplicity, unless otherwise stated, an M/M/1 system was used. This is a system with exponentially distributed inter-arrival time ($\overline{t}$), exponentially distributed average



(a) Varying inter-arrival times (b) Different replica host bandwidth

Figure 1. Average time spent in system with and without replication

service time ($\overline{x}$) in which just one server that can upload one file at a time. The model assumes just one file which is originally hosted by one peer.

4.1. Average download time for varying inter-arrival times

The Average Download Time of a file is the most significant metric for the user. In the M/M/1 system, it can be estimated as the average time spent in the system (T), which includes both queuing and service time:

$$T = \frac{1/\mu}{1-\rho}, \tag{1}$$

where ρ is the stability condition, $\rho = \frac{\lambda}{\mu}$. $\lambda = \frac{1}{\overline{t}}$ and $\mu = \frac{1}{\overline{x}}$, where λ and μ follow a Poisson distribution.

It is of our interest to better understand how the time spent in the system, T, is affected by replicating the file into another host.

First, let's see how replicating a file into another peer affects T, when both peers have the same average service time, $\overline{x} = 5\ minutes$. From Equation 1 we can obtain T_{old} and T_{new}, where T_{old} is the average time spent in the system when the file has not been replicated, and T_{new} is the average time spent in the system, when the file has been replicated into another peer, when both peers have the same average service time.

$$T_{old} = \frac{1/\mu}{1-\rho} = \frac{1/\mu}{1-\frac{\lambda}{\mu}}; \qquad T_{new} = \frac{1/\mu}{1-\rho} = \frac{1/\mu}{1-\frac{2\lambda}{\mu}}$$

Figure 1(a) shows the effect replication has on T for different average inter-arrival times ($\overline{t}$). It can be observed that T is always smaller when replication is used, but the improvement is greater when the queries are more frequent. Similar results can be observed in the simulations (see Section 6.2).

4.2. Average download time for varying up-link bandwidth

Now, let's see the effect of the peer's bandwidth on the average time spent in the system. For this case $\overline{t} = 7\ minutes$ is fixed, and $\overline{x}_1 = 6\ minutes$, which means that the peer

that originally hosts the file has a 90Kbps up-link, whereas the average service time of the peer receiving the replica, $\overline{x}_2$, varies.

$$T_{old} = \frac{1/\mu_1}{1 - \frac{\lambda}{\mu_1}}$$

$$T_{new} = f\left(\frac{1/\mu_1}{1 - \frac{f\times\lambda}{\mu_1}}\right) + (1-f)\left(\frac{1/\mu_2}{1 - \frac{(1-f)\times\lambda}{\mu_2}}\right),$$

where $f = \frac{bandwidth_1}{bandwidth_1 + bandwidth_2}$, is used to send more users to the fastest server.

Figure 1(b) shows the effect replication has on T for varying replica host up-link bandwidth. It can be observed that T is always smaller when replication is used, but the improvement is greater if the replica host has a fastest up-link.

4.3. Average download time when peers have low connectivity

To analyze how node availability affects the average time spent in the system, an M/M/m queue was be used. The abstraction is to have m servers host a file, with exponential inter-arrival times and exponential service times.

In an M/M/m system, $\rho = \frac{\lambda}{m\mu}$. The probability ϱ of all terminals being busy is:

$$\varrho = \frac{(m\rho)^m}{m!(1-\rho)}P_0,$$

where P_0 is the probability of all terminals being idle:

$$P_0 = \left[\sum_{k=0}^{m-1}\frac{(m\rho)^k}{k!} + \left(\frac{(m\rho)^m}{m!}\right)\left(\frac{1}{1-\rho}\right)\right]^{-1}$$

Then, the average time spent in system T is:

$$T = \frac{1}{\mu}\left(1 + \frac{\varrho}{m\,(1-\rho)}\right)$$

From the statistics gathered from Gnutella traces presented in [Saroiu et al. 2002b], we have that 80% of the peers participating in the file sharing system are connected to the network 50% of the time or less. To model the effect of peer connectivity on T, a single file is simulated, which is hosted by 8 peers (from the 100-peer universe) and peers have a 50% probability of not being up. The eight copies correspond to the second most popular file in the system, according to the Zipf distribution (see Section 5.2). All peers have the same average service time, $\overline{x} = 5\ minutes$.

Figure 2 shows the effect replication has on T for different average inter-arrival times $\overline{t}$, if the peers are connected to the system just 50% of the time. It can be observed that T is always smaller when replication is used, but the improvement is greater when the queries are very frequent.

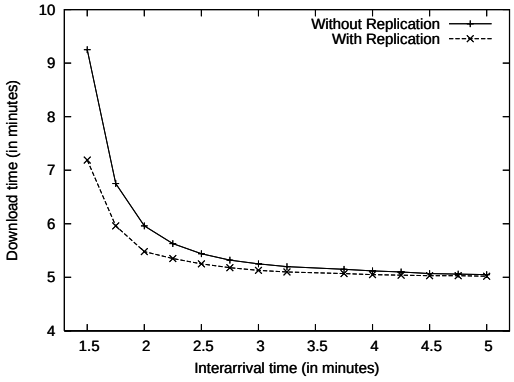
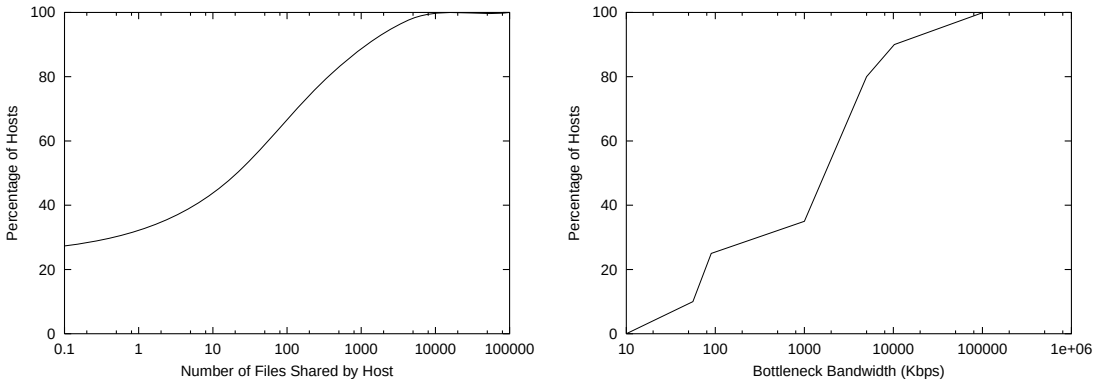


Figure 2. Average time spent in system when peers have low connectivity



(a) Shared files in simulations (b) Simulated bottleneck bandwidths

Figure 3. CDFs of different simulation parameters

5. Simulations

There are many different factors that must be simulated when dealing with P2P file sharing systems, including file distribution, file requests, peers' bandwidth, and connectivity. These factors should be simulated as closely to real life behavior as possible.

Several papers with the results of traces of the Gnutella network have been published [Saroiu et al. 2002b, Zeinalipour-Yazti and Folias 2002]. This section summarizes the most relevant results found about how P2P file sharing networks behave in real life, and indicates how this information was used in the simulations.

It should be noted that for the simulations, query distribution (file popularity) is fixed. But, as stated in [Lv et al. 2002], if one assumes that the time to complete a search is short compared to the time of change in query distribution, results obtained from these settings are still indicative of performance in real systems.

5.1. File distribution

Figure 3(a) shows the empirical cumulative distribution function (CDF) of the number of shared files per peer, used in the simulations. This CDF has been observed in real life in P2P file sharing systems [Saroiu et al. 2002b, Makosiej et al. 2004].

For simplicity, and given that the majority of shared files in current P2P file

sharing systems are music files [Leibowitz et al. 2002], all the files simulated have the same size (4MB). The same simplification has also been made in previous work in the area [Cohen and Shenker 2002, Ranganathan et al. 2002].

With respect to the popularity of files, it has been observed [Zeinalipour-Yazti and Folias 2002] that peers repeat queries frequently, especially of seasonal content (movies recently released, top 10 pop songs, etc.) so it is safe to assume that in the whole network there is a much larger number of copies of popular files than copies of non-popular files. In the initial state of the system, the number of copies of each file comes from a Zipf distribution.

5.2. Query distribution

It has been observed that queries in P2P file sharing systems follow a Zipf distribution [Cohen and Shenker 2002]. To mimic this behavior, file requests are simulated according to the popularity of files, following a Zipf distribution.

Assuming there are m files in the system, and q_i is the relative popularity of the i-th file, we have that $\sum_{i=1}^{m} q_i = 1$. In the system, each file is requested or queried according to its popularity. $P_m(i)$ is the conditional probability that, given the arrival of a file request, the arriving request is made for file i. The following formulas that define a Zipf distribution are used:

$$\Omega = \left(\sum_{i=1}^{m} \frac{1}{i} \right)^{-1} \; ; \qquad P_m(i) = \frac{\Omega}{i}$$

5.3. Bottleneck bandwidth

Figure 3(b) shows the empirical CDF used in the simulations to model the bottleneck bandwidth of the peers. The chosen CDF reflects real life bottleneck bandwidths observed in P2P file sharing systems [Saroiu et al. 2002b].

5.4. Other simulation issues

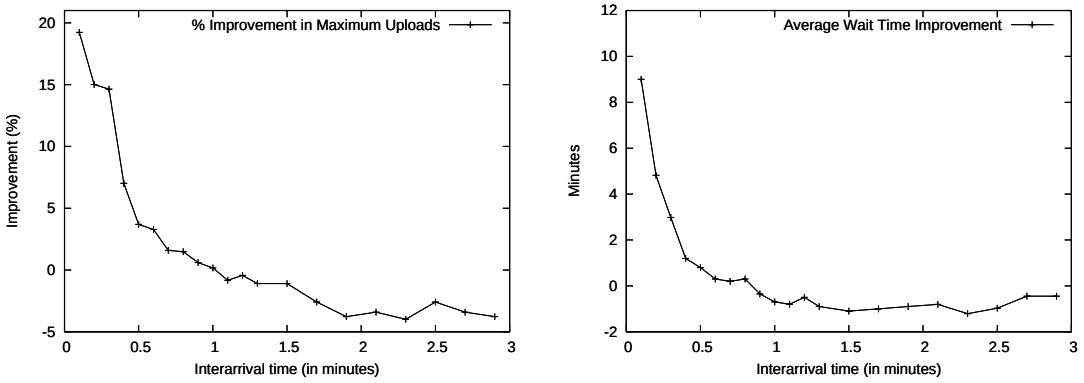
Unless otherwise indicated, the experiments simulated 100 nodes over a period of 24 hours with no extraneous traffic, across 50 runs for each case. To demonstrate that the results are not dependent on the number of files in the system, various numbers of files were simulated.

The simulations were not performed on top of a particular P2P file sharing system and the nodes have perfect knowledge of the existence of other peers, their bandwidth and the files they host. Queries are simulated with random probes, as suggested in [Cohen and Shenker 2002]. This should have no significant impact on the results since the file sharing protocol is used only for queries. Downloads are handled via direct TCP connections and are not routed through the P2P overlay.

Table 1 presents the values of the different parameters and heuristic values used in the simulations.

Parameter	Value	Description
NODES	100	Number of peers in the network
FILEFACTOR	10-100	FILEFACTOR*NODES is the number of different files simulated (there may be several copies of each file)
SIMLEN	1440	Duration of the simulation in minutes (24 hours)
THRESHOLD	2	Number of requests received by a node per interval before it decides to replicate
PERIOD	5	Interval (in minutes) for checking if threshold was exceeded
INTERARRIVAL	0.1-2.9	Query inter-arrival time (in minutes)

Table 1. Simulation Parameters



(a) Max. number of uploads by peers (b) Average download times of files

Figure 4. Improvement in load and download times, when automated replication is used

6. Evaluation

6.1. Load balancing

The metric used to measure the efficiency of the solution is the maximum number of uploads served per peer during the simulations. Figure 4(a) shows the improvement in the maximum number of uploads a peer received during the simulation period, for varying query inter-arrival times. It can be observed the best improvement is achieved when the queries are more frequent (for example, in our simulations, when queries are very frequent, the peers receive up to 20% less download requests if they are using the load balancing solution). When queries are less frequent, the improvement is negative, which means that using the proposed solution is a little bit worse than not using it. What is interesting is that for situations where queries are very frequent the proposed scheme is able to balance the upload of peers; and, in the worst case (when queries are less frequent), the peers do not suffer much from it.

The results seen in Figure 4(a) are easily explained as follows: when queries are less frequent, peers suffer from the replications but are not able to benefit much from them. On the other hand, when the queries are more frequent, the benefits from performing the automated file replication outweigh the extra-burden of replicating files and there is a significant improvement in the maximum number of uploads a peer receives during the simulation period. The results do not change if different numbers of files are simulated in the system.

6.2. Average download time

The average download time is defined as the average (for all peers) of the time elapsed since a peer requests to download a file until the time it finishes its download. Figure 4(b) shows the improvement (in minutes) on the average download times simulated for different query inter-arrival times (x-axis). The best results can be seen when the queries are very frequent. For example, when 1000 files were simulated in the system for a query inter-arrival time of 0.2 minutes, the average download time of the files was 5 minutes shorter than when the same simulation ran without replication.

From the results we can observe that when requests are less frequent, the improvement is smaller (or negative in some cases). This is an expected outcome since the results should be better when the system is more overloaded, this is, more requests are received per interval. When the system is less overloaded negative results can be observed because of the extra overhead of replicating some files that may not be used later.

7. Conclusions

This paper presented a new solution for the problem of load balancing in P2P file sharing systems, based on the automated replication of files into "good" peers to balance the load of uploading peers, improve download times and file availability. Simulation results show that for cases when the system is overloaded with file requests, a significant improvement in the number of uploads observed (load balancing) and in the average download times is observed. When the requests are less frequent, the difference is not significant.

The downside of the scheme is the extra space needed to store the replicas, but this space could be limited and new replicas could replace old –Least Frequently Used– replicas.

It should also be noted that, by using the proposed scheme, availability of unpopular files is not guaranteed to improve; but it may, in the case of peers who only host unpopular files and who may become saturated by requests for those files, and as a consequence proceed to replicate the most popular of the unpopular files.

In conclusion, using the proposed automated load balancing solution for unstructured P2P file sharing systems helps balance the load of uploading peers as well as reduce average download times when the system suffers from frequent file requests. The trade-off appears to be reasonable and current P2P file sharing systems would benefit from using the proposed scheme.

References

Adar, E. and Huberman, B. A. (2000). Free riding on Gnutella. *First Monday*, 5(10).

Bienkowski, M., Korzeniowski, M., and auf der Heide, F. M. (2005). Dynamic load balancing in distributed hash tables. In *Fourth International Workshop on Peer-to-Peer Systems (IPTPS 2005)*.

Byers, J., Considine, J., and Mitzenmacher, M. (2003). Simple load balancing for distributed hash tables. In *Second International Workshop on Peer-to-Peer Systems (IPTPS 2003)*.

Cohen, E. and Shenker, S. (2002). Replication strategies in unstructured peer-to-peer networks. In *Proceedings of ACM SIGCOMM 2002*, Pittsburgh, Pennsylvania.

Golle, P., Leyton-Brown, K., and Mironov, I. (2001). Incentives for sharing in peer-to-peer networks. In *Proceedings of the 2001 ACM Conference on Electronic Commerce (EC 2001)*.

Karger, D. R. and Ruhl, M. (2004). Simple efficient load balancing algorithms for peer-to-peer systems. In *Third International Workshop on Peer-to-Peer Systems (IPTPS 2004)*.

Leibowitz, N., Bergman, A., Ben-Shaul, R., and Shavit, A. (2002). Are file swapping networks cacheable? Characterizing p2p traffic. In *Proceedings of the 7th International Workshop on Web Content Caching and Distribution*.

Lv, Q., Cao, P., Cohen, E., Li, K., and Shenker, S. (2002). Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th annual ACM International Conference on Supercomputing*.

Makosiej, P., Sakaryan, G., and Unger, H. (2004). Measurement study of shared content and user request structure in peer-to-peer gnutella network. In *Design, Analysis, and Simulation of Distributed Systems (DASD 2004)*.

Markatos, E. P. (2002). Tracing a large-scale peer-to-peer system: An hour in the life of Gnutella. In *Proceedings of the Second IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2002)*, pages 65–74.

On, G., Schmitt, J., and Steinmetz, R. (2003). The effectiveness of realistic replication strategies on quality of availability for peer-to-peer systems. In *Third International Conference on Peer-to-Peer Computing (P2P 2003)*.

Pouwelse, J. A., Garbacki, P., Epema, D. H. J., and Sips, H. J. (2005). The BitTorrent p2p file-sharing system: Measurements and analysis. In *Proceedings of the 4th International Workshop on Peer-to-Peer Systems (IPTPS 2005), Lecture Notes in Computer Science 3640, Springer*, pages 205–216.

Ranganathan, K. and Foster, I. T. (2001). Identifying dynamic replication strategies for a high-performance data grid. In *Proceedings of the International Grid Computing Workshop*, pages 75–86.

Ranganathan, K., Iamnitchi, A., and Foster, I. (2002). Improving data availability through dynamic model-driven replication in large peer-to-peer communities. In *Proceedings of Global and Peer-to-Peer Computing in Large-Scale Distributed Systems*.

Rao, A., Lakshminarayanan, K., Surana, S., Karp, R., and Stoica, I. (2003). Load balancing in structured p2p systems. In *Second International Workshop on Peer-to-Peer Systems (IPTPS 2003)*.

Ripeanu, M., Foster, I., and Iamnitchi, A. (2002). Mapping the Gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *IEEE Internet Computing Journal Special Issue on Peer-to-Peer Networking*, 6(1).

Roussopoulos, M. and Baker, M. (2006). Practical load balancing for content requests in peer-to-peer networks. *Distributed Computing, to appear*, 18.

Sakaryan, G., Unger, H., and Lechner, U. (2004). About the value of virtual communities in p2p networks. In *Fourth International Symposium and School on Advanced Distributed Systems (ISSADS 2004)*.

Saroiu, S., Gummadi, K. P., Dunn, R. J., Gribble, S. D., and Levy, H. M. (2002a). An analysis of internet content delivery systems. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI 2002)*.

Saroiu, S., Gummadi, P. K., and Gribble, S. D. (2002b). A measurement study of peer-to-peer file sharing systems. In *Proceedings of Multimedia Computing and Networking (MMCN 2002)*.

Suri, S., Toth, C., and Zhou, Y. (2004). Uncoordinated load balancing and congestion games in p2p systems. In *Third International Workshop on Peer-to-Peer Systems (IPTPS 2004)*.

Tutschku, K. (2004). A measurement-based traffic profile of the eDonkey filesharing service. In *Proceedings of the 5th International Workshop on Passive and Active Network Measurement (PAM 2004), Lecture Notes in Computer Science 3015, Springer*, pages 12–21.

Wilcox-O'Hearn, B. (2002). Experiences deploying a large-scale emergent network. In *First International Workshop on Peer-to-Peer Systems (IPTPS 2002)*.

Zeinalipour-Yazti, D. and Folias, T. (2002). A quantitative analysis of the Gnutella network traffic. Department of Computer Science. University of California, Riverside.