Tribler: A social-based Peer-to-Peer system

J.A. Pouwelse, P. Garbacki, J. Wang, A. Bakker[1], J. Yang, A. Iosup,
D.H.J. Epema, M. Reinders, M. van Steen[1], H.J. Sips
Department of Computer Science, Delft University of Technology, the Netherlands
[1]Department of Computer Science, Vrije Universiteit, the Netherlands

ABSTRACT

Most current P2P file sharing systems treat their users as anonymous, unrelated entities, and completely disregard any social relationships between them. However, social phenomena such as friendship and the existence of communities of users with similar tastes may be well exploited in such systems, to increase their usability and performance. In this paper we present a novel social-based P2P file-sharing paradigm that exploits social phenomena by maintaining social networks and using these in content discovery, content recommendation, and downloading. Based on this paradigm's first class concepts such as taste groups, friends, and friends-of-friends, we have designed and implemented the TRIBLER P2P file-sharing system as a set of extensions to Bittorrent. We present and discuss the design of TRIBLER, and we show evidence that TRIBLER enables fast, trusted content discovery and recommendation at a low additional overhead, and a significant improvement in download performance.

1. INTRODUCTION

Traditional P2P file-sharing systems focus exclusively on technical issues, and are therefore unable to leverage the power of social phenomena. We believe that social phenomena such as friendship, trust, and a sense of community may be at least as important as technical issues, and may indeed have a large positive impact on the usability and performance of P2P file-sharing systems. For example, viewing users as *social partners*, rather then solitary *rational agents* [11], could alleviate the problem of freeriding [3], by exploiting the fact that people tend not to steal (bandwidth) from the social group they belong to.

To address this belief, we propose in this work a novel *social-based* P2P *file-sharing paradigm*, which facilitates the formation and maintenance of social networks, and exploits their social phenomena for improved content discovery, recommendation, and sharing. Our contribution is threefold. *First*, we relate the social-based P2P file-sharing paradigm to the current research challenges in P2P research (Section 2). *Second*, we present the design and implementation of TRIBLER, which materializes the social-based paradigm by adding social-based functionality to the widely popular Bittorrent system (Sections 3-6). To facilitate the formation and maintenance of social networks, TRIBLER imports existing user contacts from other social networks, e.g., MSN, and introduces permanent user identifiers (Section 4). Rather than using direct content-based searching, TRIBLER performs content discovery and recommendation based on the notion of taste buddies, that is, users with the same taste or interests (Section 5). *Third*, we show evidence that TRIBLER achieves a significant improvement in download performance, by invoking the joint effort of social peer groups (Section 6). The full TRIBLER documentation and source code is available from `http://Tribler.org`.

2. RESEARCH CHALLENGES

With current P2P file-sharing systems continuously having more than 1,000,000 users, their hidden performance and behavioral issues can be revealed only under special scrutiny. Starting in 2003, we have studied the performance of Bittorrent [14], which has been for a number of years the most popular P2P file-sharing system. Based on our, and on related studies, we formulate the following five grand research challenges for P2P file sharing, and we argue for the importance of the social-based paradigm in solving these challenges. In particular, our social-based P2P network, TRIBLER, addresses all five grand challenges.

The most difficult research challenge is the *decentralization* of the functionality of a P2P system across the various peers. Full decentralization eliminates the need for central elements in the system, which must be set up and maintained by some party and which may form serious bottlenecks, point of failures, or security threats. In particular, connecting to the network and validating accounts are difficult to implement without any central element. To date, no P2P file-sharing system exists which fully decentralizes all functionality efficiently and without loss of integrity. Social groups form a natural method to efficiently decentralize P2P systems, due to the fact that communication is mostly localized amongst group members.

The second challenge is to guarantee the *availability* of a P2P system as a whole. The operation of such a system should not depend on the availability of any particular participating peer, or of any central component, if the latter exists. Given the short availability of peers (in [14] we found less than 4% of the peers to have an uptime of over 10 hours), the availability problem is critical. Proven social incentives such as awards and social recognition could stimulate users to leave their P2P software running for longer periods, thus improving the overall availability of the network.

The third challenge is to maintain the *integrity* of the system and to achieve *trust* amongst peers. By definition, P2P systems use donated resources. However, donors cannot always be trusted, and maintaining system integrity has proven to be difficult in operational systems [7]. Data can be attacked at several levels in a P2P system, namely system information (e.g., pointers to content), meta data, and the actual content itself. This significant problem, often ignored by P2P system designers, can be solved with a social-based network, in which users actively help to clean polluted data and users can select trustworthy representatives.

The performance of a P2P system highly depends on peers donating resources. Even though the resource economy is by definition balanced (e.g., every MByte downloaded corresponds to a MByte uploaded), autonomous peers are free to decide whether to donate resources or not. Hence, *providing proper incentives* is vital to induce cooperation and to achieve

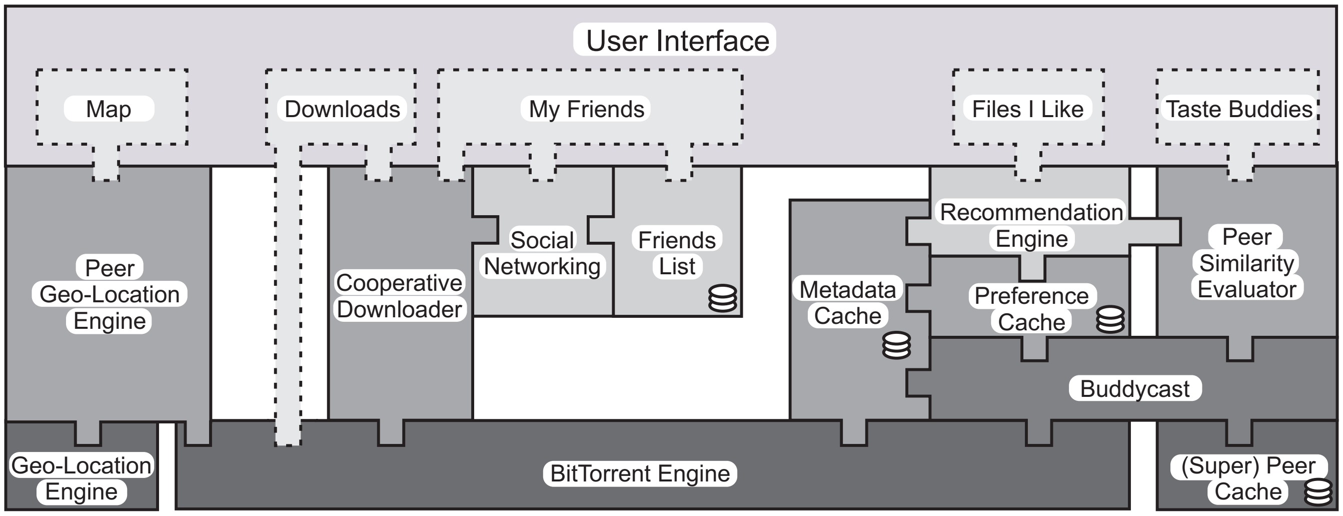

Figure 1: The system architecture of TRIBLER.

good performance [3]. Again, social recognition can help to alleviate this problem.

The fifth challenge in P2P systems is to achieve *network transparency* by solving the problems caused by dynamic IP addresses, NAT boxes, and firewalls. The fundamentals of the Internet have changed due to the wide-spread use of these three technologies. Peers no longer have the freedom to send anything anywhere, without the help of another peer acting as a mediator between them. Social networks enable communicating peers to automatically select trusted mediators from the members of their social proximity, who are still online; hence, the need for fixed mediators is eliminated.

3. ARCHITECTURE OF TRIBLER

In this section we present the architecture of our TRIBLER social-based P2P file-sharing system, which is built on top of the Bittorrent protocol. Figure 1 depicts the architecture of the TRIBLER's network client. Rectangles represent client modules. The extrusions represent *make-use-of* relationships. To achieve backwards compatibility with the existing Bittorrent network, while offering our users extended functionality, we only made modifications and extensions to the Bittorrent client software. Our system is based on the ABC open-source Bittorrent client [1]. By extending this popular client we aim to have a large users base in a relative short time, besides having a tested code base for our implementation.

Social groups The prime social phenomenon that we exploit in TRIBLER is that "kinship fosters cooperation" [12]. In other words, similar taste for content can form a foundation for an online community with altruistic behavior. In order to implement effective social groups in TRIBLER, we use an approach borrowed from evolutionary biology (see for instance [12]): we implement the ability to distinguish friend, foe, and newcomer. For this, we de-anonymize peers and facilitate social groups formation. De-anonymization is achieved by having every user choose a nickname; TRIBLER transfers user nicknames between users automatically. The *Social Networking* module in Figure 1 is responsible for storing and providing information regarding social groups (the group members, their recently used IP numbers, etc.).

Megacaches Virtually all current P2P file-sharing systems lack a persistent "memory" about previous activity in the network; peers usually exchange queries for files and file data, and completely ignore other types of information. The *context information* that needs to be saved in order to improve future performance consists of information on: social relations, altruism levels, peer uptimes, taste similarity, etc. In TRIBLER, every piece of context information is stored locally at every peer in *megacaches*, and is exchanged within social groups using epidemic protocols [9]. The small database icons in Figure 1 identify the four megacaches: the *Friends List* with information on social networks, the *(Super) Peer Cache* with information on superpeers and peers in general, the *Matadata Cache* with file metadata, and the *Preference Cache* with preference lists of other peers.

The main problem concerning megacaches is the amount of overhead traffic required to keep them up-to-date. For *Friends List* and *(Super) Peer Cache*, the cache size is below 10MB, at any given time. For the *Metadata Cache*, we have observed that in Bittorrent the number of newly injected files per day is limited to roughly 1500 [14], when *content pollution* [7] is kept to a minimum. Then, we reduced the average size of the metadata for each file to just 400 bytes using Merkle hashes [10], yielding an overhead of approximately 600 KBytes/day. For this amount of overhead, all metadata can be replicated among all peers, moving content discovery from network-based keyword searching to local metadata browsing. The *Preference Cache* is designed to store at most 10MB of data, enough for preference lists of thousands of buddies (see also Section 4).

Taste buddy-based-content discovery Locating content is critical for P2P systems. Current solutions are based on one or a combination of: query flooding, distributed hash-tables, and semantic clustering. We take a next step by connecting *people* with similar tastes called *taste buddies* instead of focusing on *files*, and by using full metadata replication.

Using the *Files I like* module (see Figure 1), each peer indicates its preference for certain files. By default, the preference list of a peer is filled with its most recent downloads. We have developed an algorithm called *Buddycast* which uses an epidemic protocol to exchange preference lists using the overlay swarm (see Bootstrapping) and which can efficiently discover a user's taste buddies (see Section 5). The *Peer Similarity Evaluator* module in Figure 1 is able to compare preference lists and determine the amount of difference in taste.

The *Recommendation Engine* module is able to compile a list of files a user most likely wants. First, each file has a metadata description containing various items. Then, a user-item rating matrix is built from the preference lists [5], and an user-based recommendation is issued by TRIBLER, based on standard collaborative filtering techniques. Last, the user interface enhances the metadata browsing experience by augmenting each file entry with the estimated interest to the user.

Downloading The *Bittorrent Engine* module in Figure 1 downloads files using a Bittorrent-compatible protocol. This module can also use the *Cooperative Downloader* module's capabilities to achieve a significant increase in file download speed, by exploiting idle upload capacity of online friends (see Section 6).

User interface The user interface is key to making a social-based network usable and, as such, a critical part of the TRIBLER architecture. The *User Interface* module from Figure 1 is split in five main components: *Map*, *Downloads*, *My Friends*, *Files I like*, and *Taste Buddies*. We have previously detailed the use of the *Downloads*, *Files I like*, and *Taste Buddies* modules. The main goal of the interface is to facilitate the formation of social groups. For this purpose, the *My Friends* module clearly displays the friends, the friends-of-friends, and the taste buddies. This *visual proximity* gives the user a more personal contact with his peers, and may help reduce asocial behavior.

Another goal of the user interface design was to ease the process of visual identification of potential collaborators. When a user is downloading a file, observed IP addresses of members of the same swarm are geo-located, then displayed on the world map, using the *Map* module. We have built a *Peer Geo-Location Engine* module on top of an freely available *Geo-Location Engine* (`http://hostip.info`).

The user interface also facilitates the use of the *Cooperative Downloader* module. The user can see which friends helped him in the past, which friends he donated bandwidth to, and friends currently online which can speedup new downloads.

Bootstrapping Finding other peers in a P2P systems after software installation is called bootstrapping. In Bittorrent, peers have to repeatedly connect to a tracker in order to discover other peers. Furthermore, the original Bittorrent protocol restricts communication to the swarm of peers that download the same file, making the bootstrapping process unnecessarily repetitive. To alleviate the bootstrapping problem, we use two mechanisms. First, a TRIBLER peer automatically uses a set of pre-known *superpeers* to bootstrap into the network, immediately after installation. The peer contacts one of the superpeers only once, upon entering the network, in order to obtain an initial list of neighbors. Second, we define a special *overlay swarm*. The overlay swarm is a swarm with no tracker, and can be used for initial bootstrapping, content discovery, and other information exchange.

Overhead analysis Table 1 shows the performance results of the TRIBLER implementation. The first test determines the number of geo-lookups a powerful test computer that acts as a superpeer can handle per second. The second test shows the performance of joining the overlay swarm. The third test is joining an overlay and validating the public key of that peer using a challenge/response algorithm. The fourth test adds the exchange of a preference list with 100 files to the third test workload. These tests show that a single peer or superpeer can handle a significant workload and that little overhead is added to the Bittorrent protocol.

Test description	Performance [ops/s]
Geo-lookup	1730
Overlay swarm join	7045
Join+challenge/response	865
Join+challenge+pref exch	844

Table 1: Performance of the TRIBLER system.

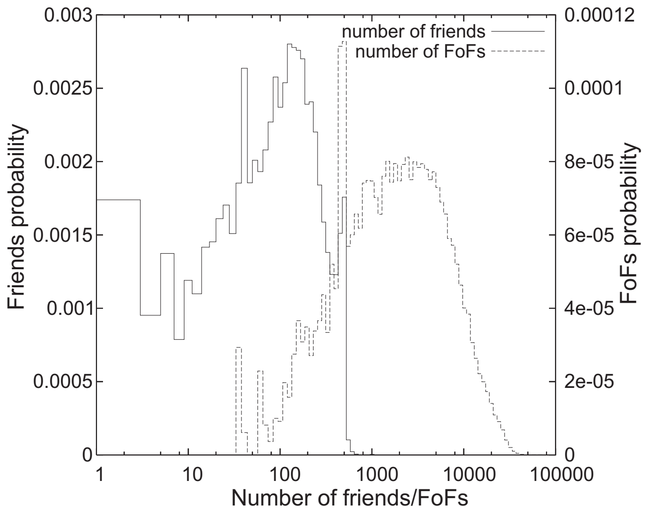

Figure 2: PDF of friends and FoFs in `Friendster.com`.

4. SOCIAL NETWORKING

A fundamental limitation in most file-sharing systems is the *session boundary*—all context information is lost when a user disconnects from the network. Due to dynamic IP numbers, it is difficult to store context information about peers across sessions. Storing long-term context information in databases like our megacaches enables the existence of trust-based social groups, but only if the identities are stable. To solve this problem, we have introduced permanent, unique, and secure peer identifiers (*PermIDs*), based on a public key scheme using elliptic-curve cryptography. To prevent peers faking the ownership of a PermID (*spoofing*) we also implemented a challenge-response mechanism for validating PermIDs. Social network creation in TRIBLER is greatly facilitated by the ability to import contacts from other social networks in which the peer is a member, e.g., MSN or GMail.

We use Bloom filters [6] for distributing and pairwise comparing the contents of the Megacaches. A Bloom filter is a very dense hash-table-like data structure for storing and (probabilistically) testing set membership. Because of their reduced size, Bloom filters can significantly reduce the bandwidth requirements of epidemic information distribution, which is the basis of our solution for content discovery and social networking.

The size of a Bloom filter depends on the number of expected connections. With the TRIBLER system not yet in full operation, we resort to analyzing another existing social network: `Friendster.com`. We have created a crawler for this network, and we have obtained 3.3 million relations between 27,000 people. Figure 2 shows the probability density of the numbers of friends and friends-of-friends in `Friendster.com`. For this data set, a person has on average 243 friends and 9,147 friends-of-friends. These figures are within an order of magnitude similar to the figures reported in [13] for several P2P file-sharing networks. Based on these numbers, we computed that 260 bytes are needed to discover the common friends-of-friends of two peers using a Bloom filter. This very low bandwidth requirement enables TRIBLER peers to exchange information simultaneously with *thousands* of other peers, which is a significant improvement over traditional epidemic protocols, and offer sufficient leverage for large-scale P2P networks.

5. BUDDYCAST ALGORITHM

In order to do content discovery and recommendation, we create in TRIBLER an overlay consisting of peers with similar tastes (taste buddies). We use the Buddycast algorithm to discover these taste buddies.

The Buddycast algorithm is based on an epidemic protocol and works as follows. Each peer maintains a list of its top-N most similar peers along with their current preference lists. Periodically, a peer connects to either (a) one of its buddies to exchange social networks and preference lists (*exploitation*), or (b) to a new peer, randomly chosen, to exchange this information (*exploration*). To maximize the exploration of the social network, every peer also maintains a list with the K most recently visited random peers, and avoids reconnecting to a peer already present in the list. In contrast to other epidemic protocols such as Newscast [9], we use both exploitation and exploration branches, we limit the randomness of peer selection during the exploration, and we implicitly cluster peers into (trusted) social groups. This is very similar with the approach proposed in [15].

To find a good balance between exploitation and exploration, the following procedure is adopted. First, $\lceil \lambda \cdot N \rceil$ random peers are chosen, where $\lambda \geq 0$ is the exploitation-to-exploration ratio. Then, these random peers are joined with the N buddies in a single ranked list, with the random peers being assigned the lowest ranks. Then, one peer is randomly chosen from this ranked list according to a roulette wheel approach (probabilities proportional to the ranks), which gives taste buddies a higher probability of being selected than the random peers. Once a peer has selected some other peer, the buddy lists of the two peers are joined. The first peer then ranks the composite list according to the preference list similarities with its own preference list, and retains only the top-N best ranked peers. Similarities between preference lists are measured using the Pearson correlation among the binary rating vector for all the known items [5].

To experimentally validate our Buddycast algorithm, we run a network of 480 peers on our DAS 400-processors system (`http://www.cs.vu.nl/das2`). We used a data set of TV watching habits of 480 users from the SKO foundation [2] as rating data (users watched or did not watch TV programs). Each peer maintained a list of 10 taste buddies ($N = 10$) and the 10 last visited peers ($K = 10$). The exploitation-to-exploration ratio λ was set to 1. Figure 3 compares the convergence of Buddycast to that of Newscast, which randomly selects peers to connect to, i.e., which corresponds to $\lambda \to \infty$. After each update we compare the list of top-N taste buddies with a pre-compiled list of top-N taste buddies generated using all data, which would be possible with a centralized approach. Figure 3 shows the fraction of overlap as a function of time (represented by the number of updates). The convergence of Buddycast is much faster than that of Newscast, and similar to the one proposed in [15]. We believe that Buddycast will perform even better for a network with millions of peers, but this kind of setup is difficult to emulate even on a large-scale system like the DAS.

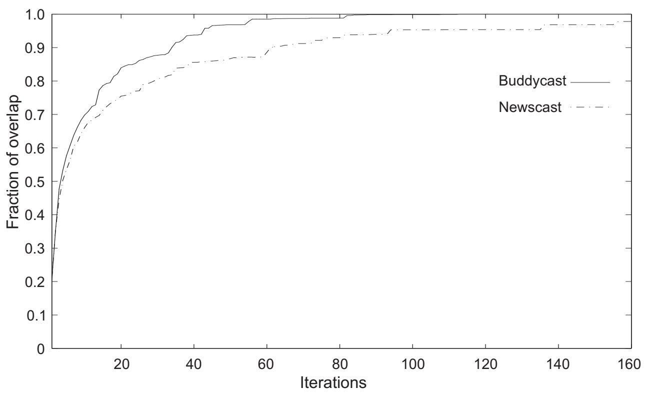

Figure 3: Convergence of our Buddycast algorithm.

6. COOPERATIVE DOWNLOADING

In this section we present the protocol based on social grouping and cooperation, which improves download efficiency. Early downloading protocols (e.g., Gnutella) have no incentives for donating upload bandwidth. This approach has serious limitations in real environments, because unconstrained bandwidth sharing is sensitive to freeriding [3]. The Bittorrent tit-for-tat mechanism was the first system which offered an incentive for uploading. The current Bittorrent mechanism also has its disadvantages, because without enough seeding peers, the download speed of a peer depends on its actual contribution to the community. In real systems this is overly restrictive, as Bittorrent's tit-for-tat bartering protocol limits a peer's effective download bandwidth to its upload link capacity. Hence, peers with asymmetric Internet access, such as ADSL or ADSL-2, cannot fully use their download capacity.

We have developed a new *cooperative downloading protocol* which makes use of social groups, where members who trust each other cooperate to improve their download performance (See Figure 4). The idea of download with the help of others was first introduced in [16], where altruistic peers contribute their bandwidth by joining a swarm even if they are not interested in the content being distributed in this swarm. The inherent assumption of sufficient altruism in the network without any incentives makes this simple approach impractical in real-world environments. Our cooperative downloading protocol solves this problem by introducing social groups incentives.

Peers from a social group that decide to participate in a cooperative download take one of two roles: they are either *collector*s or *helper*s. A collector is the peer that is interested in obtaining a complete copy of a particular file, and a helper is a peer that is recruited by a collector to assist in downloading that file. Both collector and helpers start downloading the

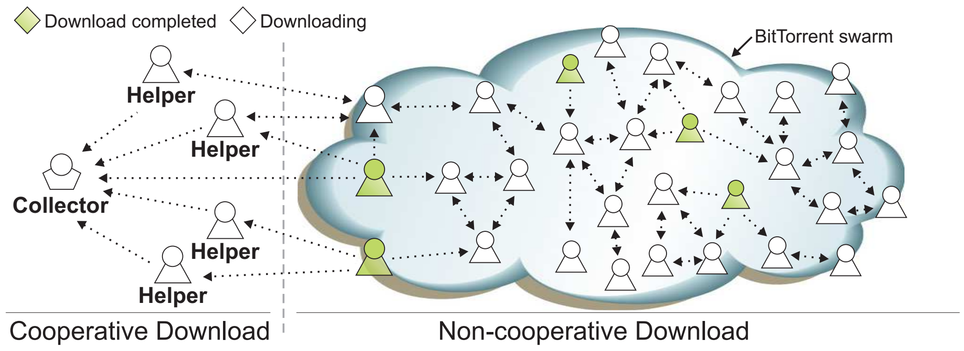

Figure 4: Overview of cooperative downloading.

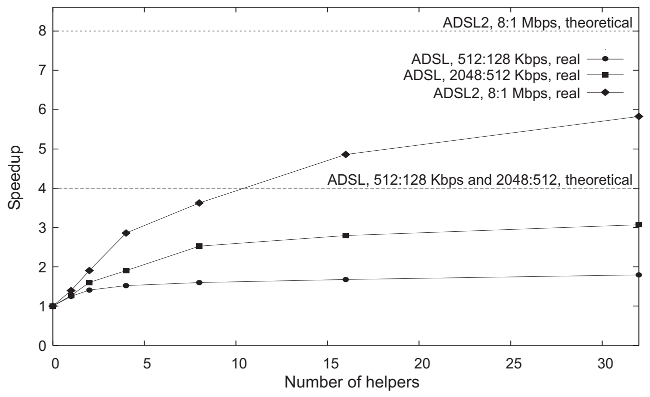

Figure 5: Helpers' influence on the download speedup.

file using the classical Bittorrent tit-for-tat and cooperative download extensions. Before downloading, a helper asks the collector what chunk it should download. After downloading a file chunk, the helper sends the chunk to the collector without requesting anything in return. In addition to receiving file chunks from its helpers, the collector also optimizes its download performance by dynamically selecting the best available data source from the set of helpers and other peers in the Bittorrent network. Helpers give priority to collector requests and are therefore preferred as data sources.

We have implemented and tested our cooperative download extension to the Bittorrent protocol in a real environment. For this we have selected a middle-sized swarm of around 1,900 peers with only 6% seeds, distributing a 1.2 GB file. These numbers remained almost unchanged during our experiments. We performed our tests for three download:upload bandwidth ratios, from standard Internet package offerings: low-end ADSL with a ratio of 512:128 Kbps, high-end ADSL with a ratio of 2048:512 Kbps, and ADSL-2 with a ratio of 8:1 Mbps.

As a performance metric of our system, we use the ratio between the download time achieved by a peer obtaining a file all by itself versus the corresponding time for a cooperative group (*speedup*). The theoretical maximum speedup, assuming that a peer's download bandwidth equals its upload bandwidth (tit-for-tat holds), is limited to the ratio between download and upload link capacities. Thus, for ADSL and ADSL-2 the maximum achievable speedup is 4 and 8, respectively.

Figure 5 shows the obtained speedups for a numbers of helpers in range from 0 to 32. The total download time was decreased with a factor of almost 2 for low-end ADSL, more than 3 for high-end ADSL and almost 6 for ADSL-2. The difference between the theoretical and achieved speedups is mainly due to influence of seeders and delays for helpers when requesting unique file chunks from peers. The more helpers are involved, the more restrictive the unique file chunk selection criterion, and consequently the longer the time needed to obtain such a chunk. This time is further increased in the case of low-end ADSL by the fact that low-end ADSL users are discriminated as those who have upload bandwidth below average [14].

7. RELATED WORK

The idea of exploiting natural connections between humans in large-scale social networks is starting to become a major research topic. To date, methods based on social clustering were applied in P2P networks to limited aspects of content distribution [8], user communities formation [4], and collaborative service provisioning [5].

In [8] a system which uses knowledge discovery techniques for overlay network creation is presented. By automatically clustering users based on their preferences, the system enables the content location and improves the performance of content sharing. In [4], a simple general-purpose system is proposed. The system groups peers based on the similarity of their keyword searches. Authors give evidence on how their system can be used to form and maintain communities of users. An extensive experimental analysis of several collaborative filtering methods is given in [5].

TRIBLER is the first system which exploits social phenomena to address all aforementioned research challenges in P2P file sharing networks.

8. CONCLUSION AND FUTURE WORK

In this paper we have presented a novel paradigm for the design of P2P file-sharing networks based on social phenomena such as friendship and trust. Following the paradigm's first-class concepts, e.g., friends, friends-of-friends, and taste buddies, we have designed and implemented the TRIBLER P2P file-sharing system. We have described how TRIBLER can help to automatically build a robust semantic and social overlay on top of Bittorrent, one of the most popular P2P file-sharing systems. We have shown how various TRIBLER components can yield good performance with respect to existing solutions. In particular, we have presented evidence that cooperative downloading can achieve double, triple, or even sextuple download speed when used in a real Bittorrent swarm. Last, but not least, we have shown how TRIBLER addresses the five major P2P research challenges.

In the mid-term future we will expand TRIBLER with a reputation system and tag-based content navigation, while in the long-term future we will incorporate application-level multicasting for video streaming.

REFERENCES

[1] `http://sf.net/projects/pingpong-abc`.

[2] `http://www.kijkonderzoek.nl`.

[3] E. Adar and B. A. Huberman. Free riding on gnutella. Technical report, Xerox PARC, August 2000.

[4] N. Borch. Social peer-to-peer for social people. In *The Int'l Conf. on Internet Technologies and Applications*, Sep 2005.

[5] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proc. of UAI*, 1998.

[6] A. Broder and M. Mitzenmacher. Network applications of bloom filters: A survey. In *40th Conference on Communication, Control, and Computing*, 2002.

[7] N. Christin, A.S. Weigand, and J. Chuang. Content availibility, pollution and poisoning. In *ACM E-Commerce Conference*. ACM, June 2005.

[8] A. Fast, D. Jensen, and B. N. Levine. Creating social networks to improve peer-to-peer networking. In *11th ACM SIGKDD*, Aug 2005.

[9] M. Jelasity and M. van Steen. Large-scale newscast computing on the Internet. Technical Report IR-503, 2002.

[10] Ralph C. Merkle. A digital signature based on a conventional encryption function. In *CRYPTO '87*, pages 369–378. Springer-Verlag, 1988.

[11] S. J. Nielson, S. A. Crosby, and D. S. Wallach. A taxonomy of rational attacks. In *4th Int'l Workshop on Peer-to-Peer Systems (IPTPS)*, Feb 2005.

[12] E. Pennisi. How did cooperative behavior evolve? *Science*, 309(5731):93+, July 2005.

[13] L. Plissonneau, J.-L. Costeux, and P. Brown. Analysis of Peer-to-Peer Traffic on ADSL. In Constantinos Dovrolis, editor, *PAM*, volume 3431 of *LNCS*, pages 69–82. Springer, 2005.

[14] J. A. Pouwelse, P. Garbacki, D. H. J. Epema, and H. J. Sips. The Bittorrent P2P file-sharing system: Measurements and analysis. In *4th Int'l Workshop on Peer-to-Peer Systems (IPTPS)*, Feb 2005.

[15] S. Voulgaris and M. van Steen. Epidemic-style management of semantic overlays for content-based searching. In *EuroPar*, Lisbon, Portugal, Aug 2005.

[16] J. Wong. Enhancing collaborative content delivery with helpers. Master's thesis, The University of British Columbia, Nov 2004.