

P2P-based PVR Recommendation using Friends, Taste Buddies and Superpeers

Johan Pouwelse, Michiel van Slobbe, Jun Wang, Marcel J.T. Reinders, Henk Sips
Faculty of Electrical Engineering, Mathematics and Computer Science,
Delft University of Technology,
Delft, The Netherlands
j.a.pouwelse@ewi.tudelft.nl

ABSTRACT

In this paper we present a novel distributed recommendation method based on exchanging similar playlists among *taste buddies*, which takes into account the limited availability of peers, lack of trust in P2P networks, and dynamic identities of peers, etc. Our preliminary experiment shows that only exchanging a small portion of similar playlists from taste buddies could lead to an efficient way to calculate recommendations within a context of P2P networks.

Keywords

Collaborative filtering, recommendation, peer-to-peer, superpeers, taste buddies

INTRODUCTION

The world of Television is finally meeting the PC world. The TV-capture card is becoming a standard PC accessory. Products from television-oriented companies now have to directly compete with products from PC-oriented companies.

Personal Video Recorder

The arrival of the Personal Video Recorder (PVR) is changing the way people watch TV. A PVR enables people to record TV programs on a hard disk. This sounds similar to a common VCR, but the selection of which program to record is much easier. A PVR uses an Electronic Program Guide (EPG) to show the user which programs are broadcasted by a TV station at a certain time. Due to the information in this EPG a PVR can be instructed to, for example, record the program *Boston Public*:

- in this timeslot at this TV station
- any time it is shown on this TV station
- every episode broadcasted on any station

Video compression techniques and ever growing storage capacity ensure that ten of hours of television can be recorded on a PVR. Studies have shown that PVR users shift from watching live TV towards watching the programs on their hard disk. After a few days or weeks of usage, PVR users become less aware of which TV station they are watching and fast-forward through commercials.

The Tivo was the first Consumer electronics (CE) based PVR in the marketplace. Hardware MPEG encoding was used to compress the video stream before storage on the hard disk. Recently, PVR-like functionality was no longer bound to CE hardware with the arrival of PC software such as MythTV¹ on Linux and the Microsoft Media Center Edition of Windows XP².

Recommendation

A recommender system can be defined as: a system which “has the effect of guiding the user in a personalized way to interesting or useful objects in a large space of possible options”[24].

A PVR recommendation system observes users TV watching habits either *explicitly* or *implicitly* in order to calculate a recommendation [2, 3, 12, 19]. A recommender system can build an *explicit* profile of a user. Such a profile is filled explicitly by the users ratings. For example, preferred channels, favorite genres, hated actors, etc. An *implicit* profile is based on passive observations and contains the TV watching habits with channel surfing actions. We believe that asking the user to rate programmes is annoying and should be avoided when possible, we therefore use implicit profiles.

A TV recommendation system can either work *stand-alone*, by using only content descriptions (content based) [2, 12], or by *collaborating* and exploiting the profiles of other users

Copyright is held by the author/owner(s).
Workshop: Beyond Personalization 2005
IUI'05, January 9, 2005, San Diego, California, USA
<http://www.cs.umn.edu/Research/GroupLens/beyond2005>

¹<http://www.mythtv.org/>

²<http://www.microsoft.com/windowsxp/mediacenter/default.msp>

(collaborative filtering based) [5, 6, 14, 17, 18, 27], or by combining both of them (hybrid) [3, 19, 24]. In this paper we focus on collaborative filtering based recommendation due to its simplicity and high quality.

Recently, a few early attempts towards decentralized collaborative filtering have been made [6, 13, 22]. Canny [6, 7] proposed a distributed EM (Expectation Maximization) update scheme. However, the update is partially distributed since a "totaler" (server) is still required to connect with all the peers. In [13], a DHTs based technique was proposed to store the user rating data. Those solutions aimed to collect rating data to apply the centralized collaborative filtering and they hold independently of any semantic structure of the networks. This inevitably increases the volume of traffic within the networks. In [30], we introduced fully distributed item-based collaborative filtering algorithm. The similarity between content (items) are derived from the profiles of the different users and stored in a distributed and incremental way as *item-based buddy tables*. By using the item-buddy tables, items are organized in the form of relevance links. Recommendation can then be done according to the similarities stored in the item-buddy tables.

Different with the above approaches, we take into account the limited availability of peers, lack of trust, and dynamic identities of peers in P2P networks. For each target user, our method selects a set of users profiles by measuring the similarity to that user. The *top-N* similar users are then identified in a fully distributed manner and their profiles are employed to make recommendations.

Peer-to-Peer

Peer-to-Peer (P2P) technology is best known for its P2P file sharing programs such as Napster, Kazaa, and Bittorrent. One definition of P2P is "a class of applications that takes advantage of resources – storage, cycles, content, human presence – available at the edges of the Internet" [26].

A general property of P2P technology is its often disruptive nature. This property arises from the fact that P2P technology eliminates central controlling points in both a technical and business sense.

In this paper we use P2P-based sharing of the detailed TV watching habits. The P2P technology must distribute this information and propagate updates. Several papers focus on a method called "epidemic distribution" to disseminate information [9, 10]. Epidemic distribution is based on forwarding information to a set of randomly selected peers. New information from a single source peer is then quickly spread out to numerous peers. This information dies out when nodes have already received this information and no longer forward it to others. This is an example of *undirected* spreading of information or gossiping without a central server.

When peers share a common interest, such as the same TV programs, it is possible to form "virtual communities" around such common interests. When common interests are

identified it is possible to spread updates of information using less bandwidth, thus in a more efficient manner.

PROBLEM DESCRIPTION

The aim of this paper is to *produce a scalable, near-zero cost solution for television content recommendation on PVR-style devices using P2P technology*. The scalability aspect is important because few recommendation algorithms scale to millions of users and millions of (archived) television programs to recommend [17]. The aim of creating a near-zero cost solution is motivated by the desire to make a PVR which is not dependant on a central recommendation server. In this paper we will show that P2P technology can replace such a central server which is difficult to scale, contains a wealth of privacy sensitive information, and may require a subscription fee. Recommendations should be calculated in a distributed fashion by the PVRs themselves. The remaining cost for generating a recommendation are then only the modest cost for communication between PVRs and PVR processor usage cost.

Given this aim, we can derive two problem domains. First the collaborative filtering algorithms and second the distribution of TV watching habits using P2P technology. Numerous publications focus on improvements of collaborative filtering algorithms. In this paper we do *not* try to improve existing collaborative filtering algorithms, but to solve the problems which emerge when the most effective algorithms are applied in a P2P setting. The problems we solved are not unique to our recommendation context, but pose a problem in the general P2P domain. We solved the following four P2P-related problem in the context of recommendations:

- Short peer uptime
- Dynamic IP addresses
- Lack of trust
- Selfish behavior

The uptime of a networked PVR is short because many people will deactivate their PVR after use to spare electricity. Each peer in the PVR network may also fail at any time due to, for example, termination of the Internet connection. This creates a severe challenge as the distributed recommender needs to have a high availability, but the underlying PVRs have *limited availability*.

The usage on the Internet of fixed IP numbers is becoming less common. The usage of dynamic IP numbers (DHCP), firewalls, and NAT boxes results in what we call the *dynamic identity problem*, it becomes impossible to directly contact or to guess the IP number of a peer which comes on-line. When all peers in a network use fixed IP numbers it is trivial to contact a previously known peer.

The intermitted on-line/off-line behavior of peers greatly amplifies the severity of the dynamic identity problem. This

amplified problem arises, for example, when two PVR devices with dynamic identities want to exchange information for a P2P recommendation algorithm on a daily basis. When these two PVRs are only simultaneously on-line for a part of the day, it is impossible for them to rendezvous again without a third peer. When the two peers are both behind a firewall, a deadlock even arises because both need to take the initiative to puncture their firewall [21]. We call this amplified problem the *rendezvous deadlock problem*.

Another issue in P2P networks is the lack of trust. The experience with P2P file sharing systems shows that maintaining system integrity while using donated resources from untrusted peers is problematic [23]. We define *integrity* as the ability to ensure that a system operates without unauthorized or unintended modification of any of its components. Calculating recommendations is vulnerable to integrity attacks. Dedicated fans could attack the integrity of the system by, for instance, spreading bogus TV watching habits on a certain TV program. The bogus information would then raise the number of people which have this program recommended. Integrity of P2P systems in general has received little attention from academia. A unique study found that for popular songs on the Kazaa P2P file sharing, up to 70 % of the different versions are polluted or simply fake [16].

Data at several levels in the system can be attacked, namely system information, meta-data level, and the raw data (content) itself. If a P2P system needs to have any integrity the following rule must be followed: *All data obtained from peers in a P2P network is by definition untrusted and must be labeled with its origin and (implicit) trust level*. It is a significant challenge to calculate recommendations and take into account trust levels.

The last problem is the selfish behavior of people. The whole concept behind P2P technology is to pool resources together and use them to deliver services. The resource economy is by definition balanced: resources are not created out of thin air. For instance, in the case of bandwidth, every download MByte has an uploaded MByte. The first challenge in a P2P network is preventing that people do not donate any resources, and are thus *freeriding* on others. In one of the first studies (August 2000) related to freeriding [1], over 35,000 Gnutella peers were followed for one day. Nearly 70 % of the peers did not contribute *any* bandwidth. We define *fairness* in a P2P system as the level to which each user gets back as many resources from the system as they donated.

To calculate distributed recommendations, each peer needs to share its TV watching habits and send regular updates of newly watched programs. Without an incentive to share this privacy, peers will freeride due to their selfish nature. The challenge is thus to create an incentive to share TV watching habits.

Year	Month	Daily TV Minutes
2002	November	190.1
2002	December	204.5
2003	Januari	216.3
2003	February	202.0
2003	March	202.1
2003	April	187.0
2003	May	173.2

Table 1: Average amount of daily television watching in The Netherlands (Source: [28]).

SOLUTION

Our solution to the above four problems is based on three ideas. First, we identify the most reliable peers in the network of PVRs and turn them into superpeers with increased responsibilities. Preferably, these nodes have a fixed IP address. This solves the reliability issues and creates stable node to solve the dynamic identities and rendezvous deadlock problem. Second, each PVR identifies the peers with similar TV taste to efficiently exchange TV watching habits (and updates). This is the method for distributing the collaborative filtering. We present simulation results showing the recall rate of our method for distribution. Third, the end-user must enter in the PVR which other PVR users are his friends to build a network of trust. By using this information we can calculate who is a friend, friend-of-a-friend, etc. and use this to information improve our second idea.

Superpeers

We first explore the problem of limited uptime in more depth before we discuss the superpeer idea. Table 1 shows the average number of minutes people were watching television in The Netherlands [28]. From these numbers we can derive a first estimate for PVR uptime, if we assume that a PVR is only online when the attached television is in use. These numbers indicate that the average peer uptime is roughly between 170 and 215 minutes, depending on the season.

Instead of merely the average uptime, it is also important to know the distribution of uptime between PVR nodes. A good source of information on the great diversity in peer uptime is from the field of P2P file sharing. We assume that the PVR uptime will show some similarity to peer uptime in P2P file sharing networks and therefore discuss it at length.

Several measurement studies of P2P file sharing networks have addressed the issues of peer availability [4, 8, 11, 25]. Most of the availability studies only span a few days [4] or weeks [8], making it difficult to draw conclusions on long-term peer behavior.

In P2P file sharing networks, a very small fraction of the peers has a high uptime (measured in weeks) because they are never turned off and have a reliable Internet connection. We believe that for networked PVR systems, a small percentage of users will leave their PVR on continuously.

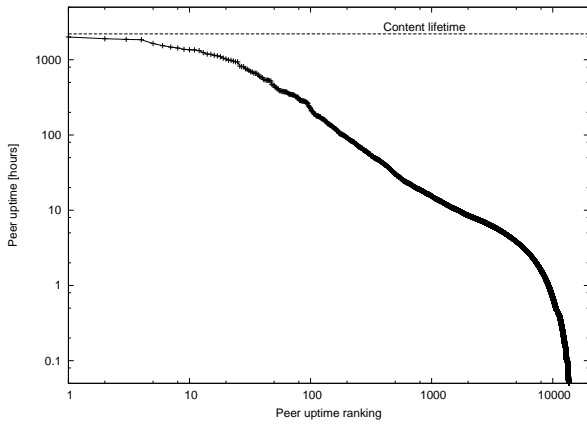


Figure 1: The uptime distribution of 53,833 peers on the BitTorrent P2P file sharing network (Source: [23]).

In a previous study the authors measured peer availability for over three months in the Bittorrent P2P file sharing system [23].

On December 10, 2003 the popular PC game “Beyond Good and Evil” from Ubisoft was injected into BitTorrent using the web site Suprnova.org and on March 11, 2004 it died. We measured the uptime of 53,883 peers which downloaded this content.

Figure 1 shows the results of our uptime measurements. Here we plot the peer uptime in hours after they have finished downloading. The horizontal axis shows the individual peers, sorted by uptime. The time scale for the uptime ranges from 3 minutes to nearly 7 months. The longest uptime is 83.5 days. Note that this log-log plot shows an almost straight line between peer 10 and peer 5,000. The sharp drop after 5,000 indicates that the majority of users disconnect from the system within a few hours after the download has been finished. This sharp drop has important implications because the actual download time of this game spans several days. Figure 1 shows that peers with a high availability are rare. Only 9,219 out of 53,883 peers (17 %) have an uptime longer than one hour after they finished downloading. For 10 hours this number has decreased to only 1,649 peers (3.1 %), and for 100 hours to a mere 183 peers (0.34 %).

From the Bittorrent measurements we conclude that a very small group of peers is significantly more reliable than the average peer. Some Bittorrent users leave their computer running for days and we assume a small percentage of PVR users will also leave their device on for days or perhaps even permanently.

We exploit this knowledge on the skewed distribution of uptime in the superpeer idea. A supernode is “a peer which has a heightened function, accumulating information from numerous other peers”, according to the definition used in the MGM studios versus Grokster case [29]. The Kazaa file sharing system uses superpeers to implement a fast file search algorithm and NAT traversal [15]. We use this con-

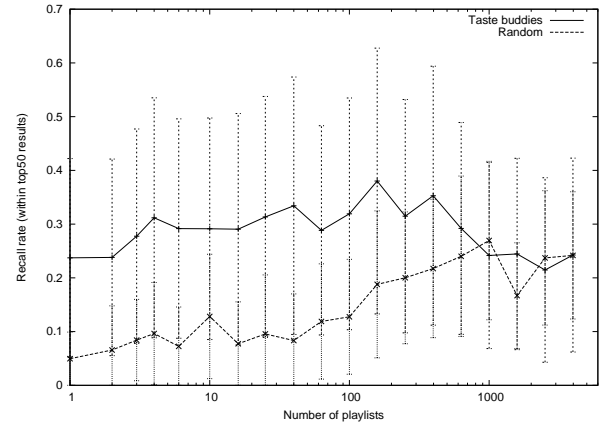


Figure 2: Comparison of the performance for random peer contacts and taste buddy contact only.

cept to distributed the recommendation.

Our solution works as follows. Each PVR keeps a large list of other peers it has seen on the network. The average uptime of each peer is used to sort this list from most reliable peer to least reliable peer. PVRs can exchange such lists and can thus discover the IP addresses of new PVRs. Note that the bandwidth requirements from exchanging such lists are modest because IP number and timestamps only require a few bytes. Each peer can request any other peer to store its current IP number. Reliable peers receive many of such requests and automatically become a reliable source of information to overcome the dynamic identity problem and the rendezvous deadlock problem. When peers come online they register their current IP number at multiple superpeers, allowing others to find them again.

Without any load balancing measures this superpeer algorithm would quickly overwhelm the most reliable peers in the network with numerous requests. We will use the friends concept to implement load balancing. Peers prefer to store their current IP number on reliable peers which belong to a friend, friend-of-a-friend, etc.

The benefit of our superpeer method is its simplicity and low overhead. It does not offer any guarantees that the new IP number of a peers becomes known to all, but in our architecture this is also not required.

Taste buddies

This section explains our solution to calculate accurate recommendations with information exchanges with just a few peers. We present simulation results which show that regular exchanges of TV habits with 100 peers is sufficient to achieve a good recommendation recall rate.

The central concept in our taste buddy idea is the exchange of TV habits which are stored in playlists. A playlist is a complete list of programs which the user has seen in the past or explicitly programmed for recording in the future. Thus, a sufficiently large number of playlists will also contain in-

formation about content which is not even broadcasted.

A simple method to exchange playlists is to contact a random peer and swap playlists. The implementation of this *random peer contact* method requires only a procedure to obtain IP addresses of peers. Due to the lack of trust in a P2P network, a minimal amount of contact with other peers is desired. Especially for the exchange of playlists because less contact lower the risk of integrity attacks.

Instead of contacting peers at random, we use a cosine similarity function [5] to identify peers with similar playlists. This ensures that we obtain playlists from our *taste buddies* only. We then apply the Amazon collaborative filtering algorithm [17] on these collected similar playlists to calculate the recommendations.

We used the MovieLens data-set [20] to evaluate the performance. We divided the MovieLens database into a training part and a testing part. We use 20 % of a users best-rated movies as the testing set. *Recall* rate is employed to measure the performance. It measures the proportion of the ground truth (GT) of the liked-items that are really recommended, which is shown as follows:

$$Recall = \frac{|liked\ items(GT) \cap recommended\ items|}{|liked\ items(GT)|} \quad (1)$$

where $|S|$ indicates the size of the set S .

The recall is calculated from the return of the Top-50 highest-ranking recommended items. We compare the recall of our taste-buddy approach to the random peer contact approach, shown in Figure 2. We varied the number of playlists which are collected to calculate a recommendation. Each data point shows the average for over 100 different runs of the recommendation. From the left to right side, playlists are exchanged from one single user to all the users, up to the size of the data-set (6,040 users).

From the figure, we observed that:

- In general, our taste-buddy approach outperforms the random peer contact approach. The performances of the two approaches converge when the playlists are fully collected.
- The recall rate of our taste-buddy approach increases with the number of the collected similar playlists increases. It reaches the peak when collecting about 130 similar playlists. Then the performance decreases as the number of the collected playlists increases. Contrarily, the recall rate of the random peer contact keeps increasing as the number of the collected playlists increases.

From these results we can conclude that only a small portion of taste buddies (similar playlists) are needed to calculate accurate recommendations. In a network of millions of PVRs, the similarity function can be used to quickly identify the peers with similar taste. We expect that this network will quickly cluster taste buddies together. The superpeer concept

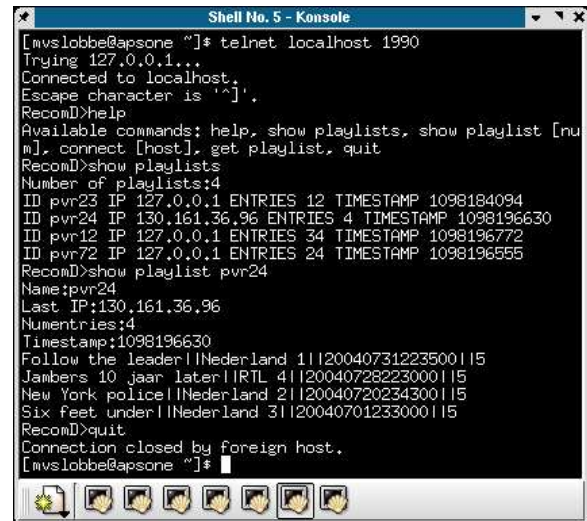


Figure 3: Debugging view of the MythTV implementation.

needs to ensure that each peer comes into contact with many peers while the similarity function will ensure the clustering.

Friends

To solve the trust issue in a P2P network, we ask the user to identify his/her real-world friends. By using social software similar to Orkut.com we can identify the social relation and distance to the users of two PVR nodes. This means that the PVR knows who your friends, friends-of-friends, etc. are on the Internet and exploit this information to ensure system integrity.

The research area of *social-aware* PVRs is still poorly understood. The idea is to set hard thresholds on the maximum social distance that a supernode user or taste buddy can be. The simple hard threshold is easy to implement and already provides a solid method to guard integrity. The concept of friends also reduces the selfish behavior, you do not freeride on your social friends.

IMPLEMENTATION

We are currently implementing all our ideas in the MythTV Open Source PVR (MythTV.org). Figure 3 depicts the inner working of this implementations. We created a daemon which is able to show and exchange playlists. This daemon has a command line interface with basic display and exchange commands.

Figure 4 shows the output of the Amazon recommendation algorithm which is now fully integrated within MythTV. We are currently looking into distributing our MythTV recommender to a large number of people to get feedback and to make some improvements.

Discussion & Conclusion

We have identified four problems when recommendations are distributed in a P2P network which are limited uptime, dynamic identities, lack of trust, and selfish behavior.



Figure 4: MythTV implementation screenshot.

By using our ideas of superpeers, taste buddies, and friends we are able to address all four problems.

We have implemented a distributed recommender using P2P technology and are currently testing its limits.

REFERENCES

1. E. Adar and B. A. Huberman. Free riding on gnutella. Technical report, Xerox PARC, August 2000.
2. L. Ardissono, A. Kobsa, and M. Maybury, editors. *Personalized Digital Television: Targeting Programs to Individual Viewers*. Kluwer Academic Publishers, 2004.
3. C. Basu, H. Hirsh, and W. W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *AAAI/IAAI*, pages 714–720, 1998.
4. R. Bhagwan, S. Savage, and G. M. Voelker. Understanding availability. In *International Workshop on Peer to Peer Systems*, Berkeley, CA, USA, February 2003.
5. J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proc. of UAI*, 1998.
6. J. Canny. Collaborative filtering with privacy via factor analysis. In *Proc. of ACM ICIR*, 1999.
7. J. Canny. Collaborative filtering with privacy. 2002.
8. J. Chu, K. Labonte, and B. Levine. Availability and locality measurements of peer-to-peer file systems. In *ITCom: Scalability and Traffic Control in IP Networks*, Boston, MA, USA, July 2002.
9. P. T. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulié. From epidemics to distributed computing. *IEEE Computer*, 2004.
10. A. J. Ganesh, A.-M. Kermarrec, and L. Massoulié. Peer-to-peer membership management for gossip-based protocols. *IEEE Trans. Comput.*, 52(2):139–149, 2003.
11. K. Gummadi, R. Dunn, S. Saroiu, S. Gribble, H. Levy, and J. Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *19-th ACM Symposium on Operating Systems Principles*, Bolton Landing, NY, USA, October 2003.
12. S. Gutta, K. Kurapati, K. P. Lee, J. Martino, J. Milanski, J. D. Schaffer, and J. Zimmerman. Tv content recommender system. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 1121–1122. AAAI Press / The MIT Press, 2000.
13. P. Han, B. Xie, F. Yang, and R. Sheng. A scalable p2p recommender system based on distributed collaborative filtering. *Expert systems with applications*, 2004.
14. T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *Proc. of IJCAI*, 1999.
15. J. Liang, R. Kumar, and K. Ross. The kazaa overlay: A measurement study. September 2004.
16. J. Liang, R. Kumar, Y. Xi, and K. Ross. Pollution in p2p file sharing systems. In *IEEE Infocom*, Miami, FL, USA, March 2005.
17. G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.
18. B. Marlin and R. S. Zemel. The multiple multiplicative factor model for collaborative filtering. In *Proc. of ICML*, 2004.
19. P. Melville, R. Mooney, and R. Nagarajan. Content-boosted collaborative filtering. In *ACM SIGIR Workshop on Recommender Systems*, Sept. 2001.
20. *MovieLens dataset*, as of 2003. <http://www.grouplens.org/data/>.
21. T. Oh-ishi, K. Sakai, T. Iwata, and A. Kurokawa. The deployment of cache servers in p2p networks for improved performance in content-delivery. In *Third International Conference on Peer-to-Peer Computing (P2P'03)*, Linkoping, Sweden, September 2003.
22. T. Oka, H. Morikawa, and T. Aoayama. Vineyard: A collaborative filtering service platform in distributed environment. In *Proc. of the IEEE/IPSJ Symposium on Applications and the Internet Workshops*, 2004.
23. J. Pouwelse, P. Garbacki, D. Epema, and H. Sips. A measurement study of the bittorrent peer-to-peer file-sharing system. Technical Report PDS-2004-007, Delft University of Technology, Apr. 2004.
24. R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 2002.
25. S. Saroiu, P. K. Gummadi, and S. D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Multimedia Computing and Networking (MMCN'02)*, San Jose, CA, USA, January 2002.
26. C. Shirky. What is p2p... and what isn't, 2000. <http://www.openp2p.com/pub/a/p2p/2000/11/24/shirky1-whatisp2p.html>.
27. L. Si and R. Jin. Flexible mixture model for collaborative filtering. In *Proc. of ICML*, 2003.
28. Stichting KijkOnderzoek. Resultaten van het onderzoek, 2004. <http://www.kijkerspanel.nl/resultaten.php>.
29. S. V. W. United States District Court, Central District of California. Mgm studios versus grokster; opinion granting def. motions for partial summary judgment, 2003. http://www.eff.org/IP/P2P/MGM_v_Grokster/.
30. J. Wang, M. J. T. Reinders, R. L. Lagendijk, and J. Pouwelse. Self-organizing distributed collaborative filtering. Submitted to WWW05, Oct 2004.