Estimating Peer Similarity using Distance of Shared Files

Yuval Shavitt, Ela Weinsberg, Udi Weinsberg
Tel-Aviv University, Israel
{shavitt,ela,udiw}@eng.tau.ac.il

***Abstract*—Peer-to-Peer (p2p) networks are used by millions of users for sharing content. As these networks become ever more popular, it becomes increasingly difficult to find useful content in the abundance of shared files. Modern p2p networks and similar social services must adopt new methods to help users efficiently locate content, and to this end approximate meta-data search and recommendation systems are utilized. However, meta-data is often missing or wrong, and recommender systems are not fitted to handle p2p networks due to inherent difficulties such as implicit ranking, noise in user generated content and the extreme dimensions and sparseness of the network.**

This paper attempts to bridge this gap by suggesting a new metric for peer similarity, which can be used to improve content search and recommendation in large scale p2p networks and semi-centralized services, such as p2p IPTV. Unlike commonly used vector distance functions, which is shown to be unfitted for p2p networks due to low overlap between peers, this work leverages a file similarity graph for estimating the similarity between peers that have little or no overlap of shared files. Using 100k peers sharing over 500k songs in the Gnutella network, we show the advantages of the proposed metric over commonly used geographical locality and vector distance measures.

1. Introduction

Peer-to-Peer (p2p) content sharing networks are used by millions of users world-wide. Searching for content is performed using search strings, which in fully distributed networks such as Gnutella [11], is propagated between peers. Alternatively, in semi-centralized networks, such as BitTorrent [13] and various IPTV and VoD [3] services, peers send their queries to a server that has enough information to find peers that hold the content. In either case, once content is located, it is downloaded directly from a selected subset of the peers.

However, current trends show increase adoption of recommendation systems for finding content, since it overcomes several limitations of traditional location of searched content. First, the abundance of content makes searching a grueling task of finding a needle in a haystack. Second, search strings are usually matched against meta-data fields (such as ID3 tags in mp3 files) that are attached to the content. Often, some of this data is missing, incorrectly spelled or encoded (such as musical genre) making it difficult for users to find the data they are looking for in the abundance of existing content (less than 10% of the queries in Gnutella network are successful in returning useful content [16]). Finally, modern services, such as IPTV often have limited user interface (like a simple TV remote control), making typing search strings extremely inconvenient for users.

Accounting for these changing needs can be accomplished using peer similarity, making it possible to find "like-minded" peers. For traditional search string propagation, peers that are similar to a searching peer are more likely to hold the searched content than other peers. In recommendation systems, it is obviously more promising to recommend content from like-minded peers.

Accurate peer similarity metrics can be beneficial for both distributed and centralized search schemes. In distributed searches, querying only like-minded peers can significantly reduce the overall query load on the network. In centralized searches, creating smaller and more accurate lists of peers that are used for each search can reduce its load. Moreover, pushing these lists to the peers themselves can improve the robustness of the network in face of server failures.

However, developing peer similarity metrics in modern p2p networks is challenging. First, in p2p networks, users do not explicitly rank their preferences, but simply download content, use it and possibly delete it or simply ignore it if they dislike it. This form of *implicit ranking* makes it difficult to assess whether users "like" or "dislike" the downloaded content.

Additionally, there is a large amount of noise that inherently exists in p2p networks, since content is mostly generated and tagged by the users. This results in an abundance of duplicate content with different titles, multiple and even conflicting tagging and spelling mistakes or ambiguities.

Finally, the abundance of peers and content creates a sparse network, having lots of users sharing lots of content, whereas each given user holds only a tiny fraction of the content, and only this downloaded content is implicitly ranked. This increases the difficulty of assessing how similar peers are.

Various searching techniques were proposed, such as approximate searching [8], semantic overlays [14] and even complete restructuring of the network [15]. However, current p2p networks employ simple string matching algorithms against files names and meta-data.

Fessant *et al.* [5] showed that there exists a "natural" clustering in p2p networks when looking at peer geographical proximity and correlation between shared content. In this work, we show that simple content correlation is not sufficient to be used as peer similarity, since the abundance of files results in poor overlap of content between arbitrary peers. For some file types, it is possible to use coarser granularity for overlap, like artists or genres in music files. However, in previous work [12], we showed that this is inaccurate, since artists or genres often fail to capture the true preferences of the users.

Recommender systems were suggested to help users find new content based on their preferences or similarity to other like-minded users. These systems have been studied extensively in recent years [10], mostly relying on the willingness of users to rank their preferences in order to provide better recommendation. However, as mentioned above, p2p networks,

alongside with new home entertainment services such as IPTV, do not enjoy the luxury of explicit ranking of content, therefore require new methods to assess peer similarity.

The primary objective of this work is to find a peer similarity metric in p2p networks, which overcomes the complexities while being efficient to calculate in fully-distributed and semi-centralized environments. Integration of the suggested similarity metric can improve accuracy and speed of searches, reduce query load [5] and improve the robustness on the network.

Overcoming the difficulties is achieved by leveraging the information about the similarity between files that are shared by peers for creating file similarity graph. Then, this information is used to accurately calculate the distance between any given peers.

Evaluation is performed on song files shared by peers in the Gnutella [11] p2p network. Collecting the songs shared by more than 1.2 million users yielded over 530k. A sample set of 100k users is used for validation and the advantages of the proposed metric over traditional metrics. Distribution of the algorithm is considered by evaluating its efficiency and applicability for real-world scenarios. Finally, we show that geographical locality should not be used as a direct indication for peer similarity, as we find diverse peer similarity values for geographically near peers.

The contribution of this work is twofold: (a) a new peer similarity metric is presented, which is simple, well suited to sparse large-scale p2p systems, efficient and robust against the existence of partial view of the network, (b) using real-world large-scale p2p network, we quantify the problems of using vector overlap, validate the applicability of the proposed metric and present problems in commonly used metrics.

2. Shared Files

Many p2p networks, such as Gnutella and KAD, employ a completely distributed approach for finding content that resides on peer storage, usually by means of flooding search queries in the network. Unless anonymity measures are used the replies are sent directly to the originating peer. Therefore the remaining peers are unaware of content that resides on other peers but was not directly sent to them. Some networks, such as Gnutella, permit content browsing, where peers can manually look at the shared folders of other peers. Other proposed networks simply propagate portions of this information in the network to allow easier location of content.

A more centralized network can have a broader view of peer content by monitoring peer activity. For example, a server storing BitTorrent trackers can monitor which peer is interested in (and probably downloading) which content. Moreover, by simply participating in many “swarms”, even without actually downloading the content, as is performed in the Apollo [13] project, can reveal a lot of information about content which is held by many peers.

In either case, it is possible to obtain information on the content that is held by peers. Assuming that n peers share overall m distinct files, it is possible to create a sparse binary $n \times m$ matrix $A(i, j)$, which indicates whether a user i shares a file j (which is a private case of the traditional collaborative filtering matrix, having only $0/1$ ranking).

Theoretically, using standard distance functions (Euclidian, correlation, etc.) between the files shared by two peers, can result in a distance value between these peers. However, in p2p networks, this simply does not work. The most significant difficulty of applying traditional vector distance functions is its extreme spareness. Even in the existence of complete information on the content that resides in all peers, which can be obtained using active crawling or centralized information, the overlap between peers is small.

This sparseness is also the result of the difficulty to identify which files are identical. Comparing the actual content of files is usually done using the MD5 hash of the files, which fails when different copies exist. Using meta-data on the other hand is susceptible to different tagging and spelling mistakes.

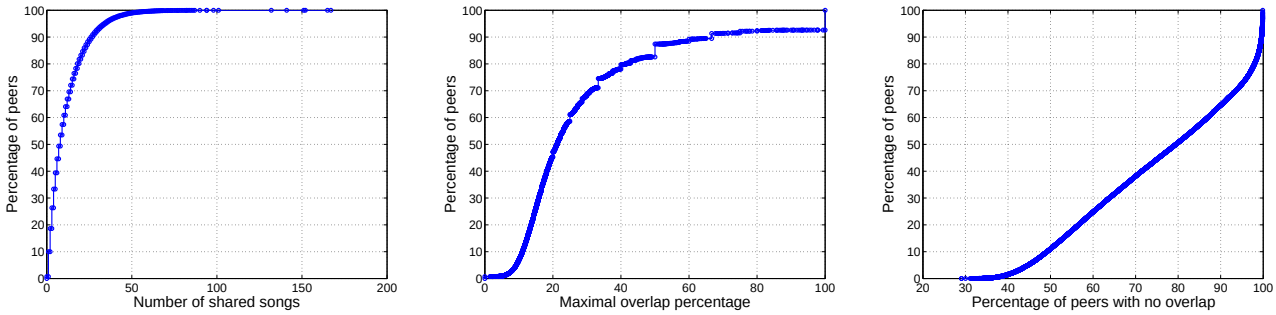
In order to illustrate the extent of this problem, we use a snapshot of the music files that were shared in the Gnutella p2p network. These were collected using a 24 hours active crawling of the shared folders of over 1.2 million peers on the 25th November 2007, selecting only files that correspond to musical content (.mp3 files). Overall 531,870 song files were collected. During the time of the crawl, Gnutella was the most popular file sharing network [9].

Identification of songs is performed using the name of the song concatenated with the name of the performing artist to account for ambiguities. We refer to this as the song id. Spelling mistakes are handled by grouping together songs that have ids with edit-distance smaller than 3, counting inserts, deletes and substations.

Using a sample set of 100k peers, we find the number of different songs each peer shares, the maximal overlap (number of songs) it has with other peers and the percentage of peers it has no overlap with. In the sampled set there are 511k songs, a value which is not much lower than the 530k songs in the original crawl using 1.2 million users. This shows that most users in the p2p network share similar files and suggests that it is not needed to perform an exhaustive crawl in order to obtain sufficient representative data.

Fig. 1(a) shows the distribution of the number of songs shared by peers in our sample. This distribution closely resembles the one reported by Zhao *et al.*[17]. Almost 85% of the peers share less than 20 songs while less than 3% share more than 50 songs, which matches the observation [1] of ”free-riders” in the Gnutella network. Also notice that all peers share less than 200 songs. We attribute this to the finite amount of disk space users are willing to devote for sharing or to the actual amount of different songs that are of interest to a user.

Fig. 1(b) shows the cumulative distribution of the maximal overlap on songs between all pairs of peers. The figure shows that 90% of the peers have a maximal overlap of 60% with at least one more peer. Moreover, 8% of peers have 100% overlap of songs with other peers. However, while this looks promising, this high overlap is mostly attributed to peers with very small number of shared songs. Furthermore, Fig. 1(c) shows the cumulative distribution of the number of peers with no overlap, revealing that 50% of the peers have zero overlap with more than 80% of the other peers. Such a high ratio of non-overlapping content between peers means that direct vector distance is unusable as a similarity measure.



(a) Shared songs (b) Max overlap (c) Zero overlap

Fig. 1. Songs shared by a sample of 100k peers, showing cumulative distributions of (a) number of shared songs, (b) the overlap of songs with the highest overlapping peer, and (c) the number of peers with no overlap

3. Peer Similarity Metric

A. File Similarity Graph

Using the file sharing matrix A, a file-similarity graph S is created. The weight of a link w_{ij} between two files i and j is the number of peers that hold adjacent files. Additionally, a popularity distribution vector C is created, counting the number of times each file appears in the network.

The similarity between files is normalized to allow comparison of similarity values between pairs of files that have different popularity. Each link weight w_{ij} is normalized using a modified cosine-distance function of the popularity of both files given in Eq. 1.

$$\widehat{w}_{ij} = \frac{w_{ij}}{\sqrt{C_i \cdot C_j}} \tag{1}$$

Since $w_{ij} \leq C_i \cdot C_j$, the normalized metric obeys $0 < \widehat{w}_{ij} \leq 1$. The normalized similarity graph is denoted by $\widehat{S}$.

We construct the similarity graph using the complete crawl of 1.2 million peers in Gnutella. During the collection of songs we only include links between songs that appear in at least 16 different peers. This helps remove "weak" ties between songs from the similarity graph, which are useless for the similarity metric. The second filter keeps, for each file, only the top 40% links (ordered by descending similarity value) and not less than 10. After these preliminary filters, roughly 20 million undirected links remain.

The degree distribution of the resulting similarity graph is shown in Fig. 2(a). The figure shows a distinct Zipf distribution with a broad set of degrees. The curve observed in the low degrees is attributed to the filtering. Similarly, Fig. 2(b) shows the number of different peers that share each song, with a clear Zipf distribution containing a long tail. This indicates that many of the songs are shared by only a few peers, proving once again the need to find a better metric than a simple file vector comparison.

B. Calculating Distance Between Peers

Once the file similarity graph is obtained, it is possible to calculate the distance between peers using all of their shared files, and not only the same files that are shared by both. The pseudo-code for the algorithm is given in Alg. 1. For any two given peers, we create a bipartite graph B that contains the songs of each peer in each side. Each peer file is connected to files of the other peer. The weight of each link is the shortest-path distance between the two files on the similarity graph (lines 3–10).

Algorithm 1 Pseudo-code of peer similarity estimation

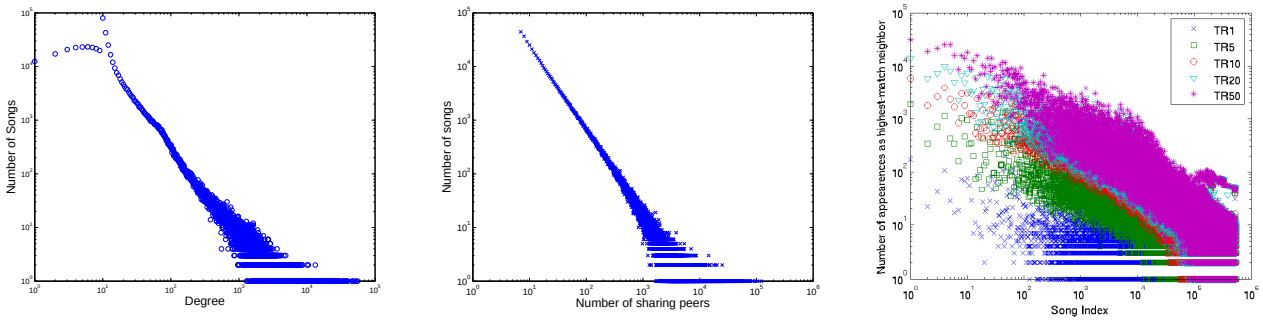
Input: Peer files matrix A, file similarity graph S
Output: Peer similarity matrix P
1: **for all** pairs of peers $(p_i, p_j) \in A$ **do**
2: $B \leftarrow \emptyset$
3: **for all** pairs of files $(f_i^k, f_j^r) \in (p_i, p_j)$ **do**
4: **if** $f_i^k = f_j^r$ **then**
5: $w = 1$
6: **else**
7: $w = d^{-1}\left(shortest\_path\left(f_i^k, f_j^r\right)\right)$, on S
8: **end if**
9: $B(f_i^k, f_j^r) \leftarrow w$
10: **end for**
11: $M = maximal\_weighted\_matching\,(B)$
12: $P(p_i, p_j) = \frac{M}{\min\{|p_i|, |p_j|\}}$
13: **end for**
14: **Return** P

In order to select the set of links and their weights, Dijkstra [4] shortest-path algorithm is executed from each of the files of one of the peers to all other files of the other peer (i.e., the target files). When the similarity graph is not fully connected, which is expected to be quite common, files that are in different components remain unconnected.

Calculating the distance between files using the file similarity graph manages to capture the "wisdom of the crowds", as it estimates the distance between files based on the global preferences of many peers. We further show how this graph is leveraged to provide the requested peer similarity, in a broader fashion than using the more traditional methods such as distance vectors or geographical proximity.

Running shortest-path on the similarity graph requires the usage of a distance function, $d(i, j)$. The distance function applies a transformation operator on all links along the shortest path between i and j, transforming the weight of the links from similarity to distances, so that two files that have high similarity value will have low distance value. When building the bipartite graph the similarity values are used as weights and not the distance values.

Once the bipartite graph is built, a *maximum weighted bipartite matching* algorithm is applied (line 11). This results



(a) Degree distribution (b) Popularity (c) Similarity sub-networks

Fig. 2. Properties of the song similarity graph, showing the Zipf distributions of the degrees and popularity, and top relation distribution of various sub-graphs

in a set of links such that no two links share a common file, and the total weight of the links in the set is maximal. In order to compare between peers i and j that have different number of files, $|p_i|$ and $|p_j|$, the weight of the matching (sum of the link weights in the matching) is normalized using $\min\{n, m\}$, which is the average link weight (line 12). This value is used as the peer similarity.

There are two main benefits of using maximum weighted matching. First, it provides an efficient method for assessing the best match between two sets of files, hence provide a good estimation for the peer similarity without requiring overlap of the shared files. Second, since only the highest similarity links are selected to the matching, various filters can be used for reducing the size of the similarity graph and improving run time and memory consumption of the algorithm, with minimal bias in the results.

C. Efficiency

The similarity of any two peers, one shares n files and the other shares m files, is calculated by running a single-source Dijkstra once and then a maximum matching on the resulting bipartite graph. Assuming that the similarity graph holds $|V|$ files and $|E|$ links, a single-source Dijkstra can be efficiently calculated [2] in $\mathcal{O}\left(|E| + |V| \ \log \ |V|\right)$.

The resulting bipartite holds at most $n + m$ files and $n \cdot m$ links. Maximum weighted matching on a bipartite graph can be computed using Hopcroft-Karp algorithm [7] in $O\left(\sqrt{n+m} \cdot (nm)\right)$. However, even less algorithms will be fast due to n and m being relatively small.

Since the most demanding phase of the similarity calculation is finding distances between files, it can be beneficial to reduce the size of the file similarity graph. Looking at the similarity graph of our dataset, reveals an extremely large and sparse graph, having less than 0.03% of possible links between songs actually exist. However, since maximum matching is robust to the removal of low-weight links, smaller sub-networks can be used. For each file, only the top N neighbors are included, ordered by non increasing normalized similarity. This extends the basic filters since it uses the *normalized* similarity values, allowing it to capture the relative popularity of adjacent files.

Using the complete dataset, we verify that these sub-networks, denoted by TRN, do not significantly change the similarity graph. For this end, the number of times each song appears as the nearest neighbor for different values of N was calculated. Fig. 2(c) shows that for N=1,5,10,20 the overall distribution is very similar in nature, and for $N \geq 10$ the distributions almost overlap. This indicates that it is possible to extract smaller sub-networks that can speed up the distance calculations, while keeping a minimal bias in the results.

To further avoid extensive traversal on the similarity graph, the shortest path procedure can be stopped once a target file is found in given distance threshold after the first target file. This is done under the assumption that further distant files are too far away to be considered relevant as a link, and hence should not affect the similarity between the peers. Files that do not have a link are removed from the bipartite graph.

Notice, however, that stopping shortest path before reaching all targets causes the proposed algorithm to be not symmetric to the selection of the peers. The peer that is the base for the shortest-paths will have all files includes, while the latter may have some files that are removed from the bipartite graph since no link reaches them. However, links that are added from one peer but are not added from the second peer are unlikely to be selected in the maximum matching, since these are mostly low-similarity links.

D. Applicability

To be truly effective in p2p networks, the similarity metric should be integrated with the p2p network and seamlessly provide users improved search results. In semi-centralized networks, that consist of servers that help locate content, all the functionality can be easily included in the server.

However, in completely distributed networks, like Gnutella network, this can be achieved by deploying a set of ultra-peers (or “hubs”) that capture a large percentage of search queries [6]. Partial information can be shared amongst ultra-peers in order to have replications of the similarity graph, hence improving its redundancy and scalability.

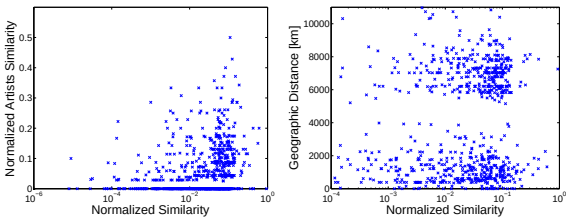
Another aspect is the highly dynamic nature of p2p networks, where files are constantly added and removed by peers. Removing files is mostly done by peers that want to preserve local disk storage or upload bandwidth, thus they do not reflect a change in the true similarity between files. However, there are peers that delete files when they are not satisfied with what they have downloaded. This means that the deleted file should

not be related to the other files which are co-shared by the peer. We assume that files, which are downloaded by users by mistake, are relatively rare or at least not highly correlated between different peers, and therefore their corresponding links usually have very low weights, which are filtered and therefore do not have an affect on the similarity between peers.

On the other hand, adding files to the network changes the similarity graph, where each file contributes a single vertex but can lead to many different links between it and the additional peer shared files. However, only when the new file is shared by many users that have common files, its links with the other files become significant enough to be included in the similarity graph causing for a change in the similarity graph.

4. Validation

Validation of the similarity metric is performed using the sampled set of 100k peers, TR10 similarity graph, a distance function $d(x) = -\log_2(x)$ and a cut-off of 1.5 in the shortest-path process. For each pair of peers, we compare the resulting similarity to the artist similarity. For this end, the artists performing the songs shared by each peer are resolved, using the ID3 tags in each file. Assuming that two peers i and j have two sets of artists A_i and A_j, the artist similarity is defined as $(|A_i \cap A_j|) / \min\{|A_i|, |A_j|\}$.



(a) Artists (b) Geography

Fig. 3. Comparing peer similarity with geographical distance and artists similarity (500 random peers are shown for brevity)

Fig. 3(a) shows that high peer similarity indicates high artist similarity, which validates the overall correctness of the similarity metric. However, high peer similarity also exists when artist similarity is zero, showing that comparing exact songs between peers, even using coarser granularity is insufficient.

We further wish to examine the correlation between files and geographical distance of peers [5], by considering the geographical distance between peers and their corresponding similarity. Fig. 3(b) compares the geographic distance of peers from the dataset to other randomly selected peers, and shows the advantage of using the proposed similarity over similarity based of geography. Although it is possible to see that there are clusters of peers that share similar geographical distance and similarity values, for roughly the same distance there is a wide range of different similarity values. As such, it is possible to see that there are peers with high similarity that are both geographically far and near.

These observations suggest that while locality of interest may still be valid in today's p2p networks, similarity of files shared by peers (or at least music preferences, which are studied in this evaluation) takes a more global approach. As such, bootstrapping peers or creating a semantic overlay of peer links based solely on shared content correlation or geographic locality is insufficient. Adding the proposed peer similarity metric manages to include a more global "wisdom of the crowds" into the process of peer similarity estimation.

5. Conclusion

This paper presents a new metric for peer similarity, based on the distance between shared files on the file similarity graph. This metric is well suited to large-scale p2p networks, where the overlap of files shared by peers is low, mostly due to the vast sparsity of the network.

As peer-based networks and services become ever more popular, it is important to find new ways for improving the ability to find useful content in the network. Better estimating the distance between peers is an important building block in most search paradigms, such as approximate search and recommendation systems, and it can help make semi-centralized services more robust to failures.

Acknowledgement This work was partially funded by the Israeli Science Foundation's center of knowledge grant 1685/07 and by the IBM 2008 Faculty Award.

References

[1] E. Adar and B. A. Huberman. Free riding on Gnutella. *First Monday*, Sept. 2000.

[2] M. Barbehenn. A note on the complexity of Dijkstra's algorithm for graphs with weighted vertices. *IEEE Transactions on Computers*, 47(2):263, 1998.

[3] M. Cha, P. Rodriguez, S. Moon, and J. Crowcroft. On next-generation telco-managed P2P TV architectures. In *IPTPS*, 2008.

[4] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[5] F. L. Fessant, A. M. Kermarrec, and L. Massoulie. Clustering in peer-to-peer file sharing workloads. In *IPTPS*, 2004.

[6] A. S. Gish, Y. Shavitt, and T. Tankel. Geographical statistics and characteristics of p2p query strings. In *IPTPS*, 2007.

[7] J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, 1973.

[8] G. Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33:2001, 2001.

[9] Ars Technica Report on P2P File Sharing Client Market Share, http://arstechnica.com/old/content/2008/04/study-bittorren-sees-big-growth-limewire-still-1-p2p-app.ars.

[10] P. Resnick and H. R. Varian. Recommender systems. *Communications of the ACM*, 40(3), 1997.

[11] M. Ripeanu. Peer-to-peer architecture case study: Gnutella network. *IEEE International Conference on Peer-to-Peer Computing*, 2001.

[12] Y. Shavitt and U. Weinsberg. Song clustering using peer-to-peer co-occurrences. In *ISM '09: Proceedings of the 2009 11th IEEE International Symposium on Multimedia*, 2009.

[13] G. Siganos, J. Pujol, and P. Rodriguez. Monitoring the Bittorrent monitors: A bird's eye view. In *PAM*, 2009.

[14] S. Voulgaris, A.-M. Kermarrec, L. Massoulié, and M. van Steen. Exploiting semantic proximity in peer-to-peer content searching. In *10th International Workshop on Future Trends in Distributed Computing Systems (FTDCS 2004)*, China, May 2004.

[15] B. Wong, Y. Vigfússon, and E. G. Sirer. Hyperspaces for object clustering and approximate matching in peer-to-peer overlays. In *USENIX HOTOS '07*, pages 1–6, Berkeley, CA, USA, 2007. USENIX.

[16] M. A. Zaharia, A. Chandel, S. Saroiu, and S. Keshav. Finding content in file-sharing networks when you can't even spell. In *IPTPS*, 2007.

[17] S. Zhao, D. Stutzbach, and R. Rejaie. Characterizing files in the modern Gnutella network: A measurement study. In *SPIE/ACM Multimedia Computing and Networking*, 2006.