Taxonomy Caching: A Scalable Low-Cost Mechanism for Indexing Remote Contents in Peer-to-Peer Systems

Kjetil Nørvåg[1], Christos Doulkeridis[2], and Michalis Vazirgiannis[2]
[1]Dept. of Computer Science, NTNU, Trondheim, Norway
[2]Dept. of Informatics, AUEB, Athens, Greece
Kjetil.Norvag@idi.ntnu.no, {cdoulk,mvazirg}@aueb.gr

Abstract

High storage capacity and support for wireless internet is now available in an increasingly higher number of mobile devices. These devices can be connected in a P2P network, thus enabling sharing of resources *(which can be both files and services) with other users participating in the network. An important challenge is to enable capabilities for finding relevant resources stored at other devices. In this paper, we present an approach to improve P2P search that is particularly suitable for connections with limited bandwidth, as in the case of portable devices. The contents of a peer are represented by taxonomy terms, and remote peers maintain a taxonomy overview instead of detailed indexing information of remote peers' contents. We show through simulations that our approach, while achieving comparable recall compared to basic flooding, significantly reduces 1) the number of messages needed for performing a query and 2) the number of peers that have to be contacted during the query.*

1. Introduction

High storage capacity and support for wireless internet is now available in an increasing number of mobile devices. Frequently these devices contain multimedia documents (including both plain text files, pictures, music, and videos) that can be shared with other users. An important challenge is to make it possible for users to find contents stored at other devices. Because of the dynamics of mobile devices, a P2P architecture is very suitable for connecting the devices because of high degree of autonomy and low administration cost. P2P systems are also intrinsically failure tolerant. We assume that the devices in the system either provide sharable files or services providing contents (which can be also files). In our previous work, in the context of MobiShare [7, 18], we have demonstrated how mobile devices are capable of sharing their resources, by hosting web services. These web services are simple data sharing services or more advanced applications communicating through web services. We will in the following denote available files and services as *resources*.

One of the main challenges in a P2P system is to provide efficient algorithms for search and retrieval of contents stored on remote peers. For non-portable computers this problem can be solved by having the data available as web sites, and search engines regularly crawling the sites and providing a searchable index. However, search engines are not applicable for mobile devices: documents stored on these devices can be assumed to be both dynamic and volatile, and the device itself might be available on a particular position in the P2P network for only a limited amount of time. Moreover, a search engine approach based on crawling documents on mobile devices is not feasible because of bandwidth restrictions. Instead, some P2P search algorithm is used. At one extreme is the baseline search technique, flooding, which is simple and robust but imposes high cost, except for the case of very small networks. On the other extreme is indexing using DHTs (also called structured P2P systems), which provides efficient searching, but suffers a high maintenance cost in the case of high churn rate, which is the case with mobile devices. In between these two extremes we find techniques applicable to unstructured P2P networks that increase the probability of finding contents or reducing the search cost. Examples of such techniques include routing indices that indicate which direction (to which neighbor peer) a query should be directed to, maintaining summaries of contents of remote peers (for example using a variant of bloom filters), as well as caching results of previous queries.

Full-text indexing of local contents at peers has a prohibitive cost in mobile environments, and is also unsuitable for multimedia files like music and video. One of the typical solutions to this problem has been to use descriptive terms in filenames, however this limits the success of a search to users that know what terms to expect. A better approach is to assume *files are classified as belonging to one or more categories of a taxonomy* (the taxonomy does not have to

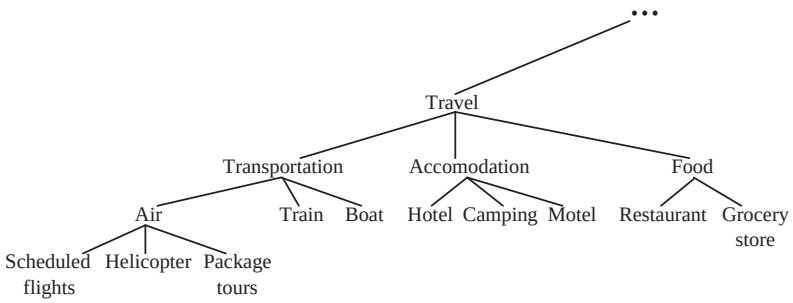

Figure 1. Part of example taxonomy.

be the same for all users). A simple example taxonomy is illustrated in Figure 1. An important feature of a taxonomy is that an object belonging to a certain category is also implicitly classified to all super-categories (i.e., ancestors) in the taxonomy tree.

Our novel approach to increase retrieval quality and reduce search cost in P2P systems, is to use summary indexing, where instead of maintaining a detailed index of the contents of remote peers, the contents of a peer are represented by terms in a taxonomy, and remote peers maintain a taxonomy overview of remote contents. We call this approach *taxonomy caching*. Representing resources with taxonomy concepts has the following advantages:

- Maintenance cost is reduced in terms of network bandwidth usage because less information has to be transferred. This is especially important in the case of mobile devices.
- For many applications a taxonomy description is more applicable.
- Taxonomy description is very useful for multimedia content, where there might be no text available for creating text-based summaries.

In this paper, we look into taxonomy descriptions, and how these can be cached in order to reduce to a minimum the amount of information that has to be transferred. An important observation is that it is not only leaf nodes in a taxonomy tree that are cached, it could also be concepts [1] on a higher level in the tree, which covers a number of nodes below. As will be described later in this paper, by employing the taxonomy caching technique, the query horizon of P2P queries will be extended, thus increasing the probability of finding information at distant peers.

The main contribution of this paper is the use of taxonomy caching towards reducing search cost and improving search quality (i.e., number of files/documents that can be found). We also study the cost and quality of the approach based on simulation experiments.

The organization of the rest of this paper is as follows. In Section 2, we give an overview of related work. In Section 3, we describe taxonomy-based routing and the architecture and use of the TCache. In Section 4, we provide an evaluation of our approach based on a simulator prototype of a P2P system utilizing taxonomy caching. Finally, in Section 5, we conclude the paper and outlines issues for further work.

2. Related work

Basic search techniques in unstructured P2P systems include flooding and random walk strategies. For an analysis of these techniques we refer to [8].

In order to reduce the cost of search in unstructured P2P systems, a number of techniques have been proposed, including: 1) techniques for improved routing that give a direction towards the requested resource, examples of such techniques include variants of routing indices [3, 14, 21, 22], use of Bloom-filters [5, 9], and connectivity-based clustering that creates topological clusters that can be used as starting points for flooding [15]. Routing indices only use higher-level metadata or summary data, but it is also possible to cache more of the contents of neighbor nodes, for example using one-hop replication [2].

Taxonomies have been used for a long time for content description. In the context of improved web search, search engines like Yahoo! and Google provide directories which make it possible to browse the web by topic.[2]

The two most relevant approaches using taxonomies to describe contents and improve efficiency of searching, are the work by Pireddu and Nascimento [14] and the work by Löser [10].

1 We use the terms ”term”, ”concept” and ”category” w.r.t. the taxonomy interchangeably.

2 See e.g., `http://dir.yahoo.com/` and `http://www.google.com/dirhp`.

Pireddu and Nascimento [14] present a generalization of routing indices [3]. The contents of peers are partitioned into categories of a predefined taxonomy, and for a particular peer the number of documents for each node in the taxonomy is known. In order to improve search and avoid flooding, taxonomy-based routing indices are employed: a peer knows also 1) the contents and number of documents of each taxonomic node of its neighbor peers, 2) the contents and number of documents of each taxonomic node on all but the lowest taxonomy level of the peers two hops away, 3) the contents and number of documents of each taxonomic nodes on all but the 2 lowest taxonomy levels of peers 3 hops away, etc. Thus a peer has much knowledge of neighboring peers' contents, but less knowledge of remote peers' contents. This information is used to route queries in the direction where the probability of finding suitable contents is maximized, while the information about contents is distributed by update messages. Their experiments show that the approach can improve search for peers containing contents of a particular taxonomy. Our work differs from the work of Pireddu and Nascimento by providing adaptive caching of remote taxonomies and also providing information about contents on distant peers and thus providing the potential of *jumps*, instead of only neighbor node routing. This increases the possibility of finding rare and/or remote contents.

In [10] Löser describes a structured P2P system where contents are classified according to a taxonomy. The description is represented as a taxonomy path, and the possible prefixes of the path are stored in a DHT. This makes it possible to perform queries on both exact description as well as generalized descriptions. This differs from our approach by assuming a separate and *structured* overlay network for storing the information.

A number of P2P projects have used RDF to describe contents and make more powerful querying on metadata possible. One example is Edutella [12], which implements a RDF infrastructure on top of JXTA. Another example is Saglio et al. [16] that describes how to query distributed RDF knowledge bases in a P2P network, but their focus is on query processing, query rewriting, and merging of results. In both approaches there are no means of query routing, except for what is already supported by the system (flooding) or maintaining routing knowledge outside the peers where the data is stored.

One issue when having taxonomy-described contents is the use of different taxonomies. In [17] Tzitzikas proposes the use of taxonomies to describe contents on peers and mediators to provide searching in the context of different taxonomies. The techniques in [17] are not P2P specific and the taxonomies are not used for query routing.

Our approach also has similarities with gossiping [5] and epidemic-style protocols [19], in the sense that information about contents of other peers is gradually disseminated to the P2P network, under the assumption that contents are sufficiently static, so the distributed knowledge will converge. A major difference compared to our approach is that dissemination in our approach is more focused, i.e., it is based on requests. Because it is summary information it also tolerates dynamic contents better, and furthermore the adaptive choice of accuracy in the cached taxonomies also makes maintenance cheaper and robust.

It should be mentioned that the use of taxonomy caching also has similarities to other kind of overlays that are created to improve searching. One such class of overlays is *semantic overlays* [4, 11, 6], which aim at connecting peers that store similar contents. In that way searching can be performed very efficiently, as soon as one of the peers containing relevant content is found.

Finally, our approach is also related to caching in general. In the context of unstructured P2P systems previous research has focused on result caching [1, 13, 20]. Result caching is orthogonal to our work, and could be applied in addition to taxonomy caching.

3. Taxonomy tree caching

The basic idea behind taxonomy caching is 1) having the contents that are stored on the nodes described according to taxonomy categories, 2) having the information about contents (i.e., categories stored on a peer) cached at remote peers in a *taxonomy cache* (TCache), and 3) using this knowledge about content at remote peers to route queries to the appropriate peers. By using the information in the TCache for routing queries, 1) the query latency can be reduced, 2) the total search cost can be reduced because a smaller number of nodes have to be accessed in order to retrieve sufficient number of results, and maybe most important: 3) because the taxonomic information may be distributed to very distant peers, the effect of the limited query horizon normally associated with search in unstructured P2P systems can be avoided. Through the adaptive granularity of contents in the taxonomy cache, the maintenance cost can be kept at a low level.

In this section, we first describe the basic concepts regarding taxonomy-based querying, and then the taxonomy-based routing and the architecture and use of the TCache are described.

3.1. Taxonomy-based querying

All globally searchable resources R_i (documents, images, videos, services, etc.) are classified with respect to one or more categories C_j (for example *Train* or *Boat* in the taxonomy of Figure 1) of a taxonomy T_k. Note that a resource classified to a category C_j is also implicitly classified to all

super-categories (i.e., ancestors) of C_j in the taxonomy tree. In general, a resource may belong to more than one category, so the taxonomy information of a resource is a tuple consisting of resource identifier, taxonomy, and a set of categories: $(R_i, T_j, \{C_j\})$.

A query Q_i is either a request for all contents belonging to a category C_q, or a request for all contents belonging to a category C_q *and* satisfying some additional property, for example textual contents (keyword-search) or metadata property.

We assume the P2P network is unstructured (Gnutella-like), although it should be mentioned that the taxonomy caching can also be useful in a structured P2P systems employing DHTs (Chord, CAN, etc.). Each peer has an identifier P_i, typically composed of the IP address and port number.

In the basic case, when no caching is employed, querying is performed as follows: the query Q_i originating from the *querying peer* is forwarded to other peers, denoted *remote peers*. The process of deciding to which peer(s) to forward the query is called *query routing*. Attached to the query is a time-to-live (TTL) value which is initialized by the querying peer. The TTL is decremented each time the query is forwarded, and when it reaches zero the query is not propagated further.

The basic query routing algorithm can be simply flooding or random forwarding, possibly employing more sophisticated techniques like routing indices. The peers that can be reached at a particular time from a peer constitute the *query horizon*.

All remote peers receiving the query execute the query locally and forward it to one or more neighbors. Local query execution is performed by matching the taxonomy-based query (and additional predicates if applicable) with local content, and if a match is found, the resource name/metadata is returned to the querying peer.

3.2. Taxonomy caching

After receiving the query results from a number of peers, the querying peer can cache the *query results* for more efficient processing of future queries as described in, e.g., [1, 20, 13]. An expiration time (presumably with a low value) would be attached to the result, to ensure that the results will be discarded after a certain amount of time. In particular for static contents and stable peers, the result caching technique can significantly reduce the search cost. However:

- The results might soon get invalid/stale, and caching the results might incur a considerable storage cost. Thus the results can only be kept for a very limited amount of time, limiting the usefulness of the caching.
- The stored results are only useful when the *exact* same query is issued several times.
- The result caching in itself will not improve the probability of finding contents outside the query horizon (note that this can be improved on, by allowing query results to be used also for queries from remote nodes, i.e., not only for local query processing).

An alternative to result caching is to instead cache taxonomical description of the contents located at remote peers in a *taxonomy cache* (TCache). This taxonomical information (i.e., the categories the results belong to) can be returned together with the query results. Caching taxonomical information has a number of important advantages:

- It is a very compact representation.
- It is more robust to changes than result caching:
  - Even though the contents of peers change, the updated contents will often belong to the same categories as before.
  - New resources that are added will often belong to the same category as existing resources, thus increasing the probability of finding them during a search, as well as not requiring additional storage for the remotely cached category.
- It can be employed in combination with result caching. This is in particular useful for very frequent queries, which then only have to be reissued at regular times.
- By using the TCache for query routing the query horizon is extended in a more robust way than when using result caching. It will also gradually be extended with time as taxonomic information is further distributed.

We will now describe query routing using the TCache, how taxonomic information is distributed, and the architecture of the TCache.

3.2.1. Query routing using the taxonomy cache. Query routing is the process of deciding where to forward a query, performed by all peers that are involved in query forwarding, i.e., both the query peer as well as intermediate peers. In both cases, when a lookup in the TCache for the category C_q in the query gives as result a matching category and the peers containing the resource, the query can be forwarded directly to one or more of these peers. This forwarding is called a *jump*. Note that unlike routing indices that only maintain information of the neighborhood and are used to choose appropriate neighbor peers for query forwarding, information about contents at very remote peers (also beyond the query horizon) can be contained in the taxonomy cache. In the case when a peer does not find a match in the TCache, the query is forwarded using the basic query routing algorithm, e.g., flooding or random walk. Note that even if a query match is found at a peer and results returned to

the query peer, the query is further forwarded until the TTL reaches zero.

It might be the case that the TCache has more than one entry for the category C_q (let's assume c entries). In this case the query is forwarded to k of these c peers. When $k < c$ a decision has to be made to which of the peers to forward the query. Our approach is to rank the candidate peers based on the number of resources they contain for the category C_q. Those that have the highest number are considered *experts* on the topic and are more appropriate to answer the query. When forwarding the query there is also the issue on how much the TTL should be decremented. If query forwarding is based on flooding, the TTL is simply decremented by one, as in the case of basic flooding. However, in the case of random walk where a query is normally only forwarded to one node, we follow a different approach: the query forwarded to the higher ranked peer is given a TTL decremented by one, the other peers are given a TTL with value zero so that they only evaluate the query but do not forward it further afterwards.

It is possible to combine the use of routing by means of taxonomic information with one of the basic searching techniques. This prevents a query from being forwarded to a distant peer without searching the peers in the neighborhood. The reason for this is that it is often the case that neighbor peers are also related to the query peer and they have contents relevant to the query. One example is peers belonging to the same university and queries for university-related information. In order to solve this problem two different queries are issued from the query peer: one that is allowed to use taxonomic information for routing, and one that should only use the basic searching technique (and hence limited by the TTLvalue).

3.2.2. Distributing taxonomic information. The basic mechanism for distributing taxonomic information is by piggybacking a matching category with the result of a query. Note that in the case a query is for a non-leaf category C_k, the query will also match sub-categories of C_k. In this case the actual categories of the results are explicitly returned in the query.

The query is returned through the return path, possibly involving jumps. The reason for returning this information through the return path is to make it possible for the intermediate nodes to know that their routing information is still valid. In particular jumps are performed because the peer from which the jump is initiated has information about the destination peer containing resources related to the category that is searched. This information is in the TCache and has a validity time (see below). By routing the information back, the peer can reinitialize the time counter (because it knows the entry is valid). It is also possible to return a message indicating that the destination peer does not have any information in the category anymore, if that is the case.

3.2.3. Architecture of the TCache. In general, a search in the taxonomy cache is a lookup in order to find all known peers containing contents of a particular category C_q (which might include the sub-categories of C_q).

As a starting point, we assume a taxonomy tree, where each node in the tree has associated a list of peer identifiers containing resources, classified as belonging to the category of the taxonomy tree node. Attached to the peer identifier is also a TTL counter. At regular times all counters in the tree are decremented, and when a counter is zero the peer identifier is removed from the tree. If a node has no more peer identifiers attached and has no children, the category node is removed from the cache.

When a category C_i is discovered at a remote peer P_i, the peer is included in the peer list of the appropriate category. If the category C_i is not already in the tree, it is inserted into the tree. In addition to the peer identifier and the TTL, also the total number of resources the peer keeps in the particular category is stored. In this way it is possible to know whether the peer is an *expert* on the topic. This can aid in selecting peers at query time.

3.2.4. TCache maintenance. One possible problem of the taxonomy tree described so far, is that it might grow very large. In particular, this might happen in the case of peers containing resources of many categories, for each category the peer identifier will be stored in the tree. Although this problem could be solved by using small TTL values in the tree, this would significantly reduce the usefulness of the tree. A better approach is to take advantage of the fact that for peers that contain many resources in many different categories, it is frequently the case that these categories can be compacted into a smaller number of super-categories. For example, assuming the taxonomy in Figure 1, if a peer identifier P_i is stored in both nodes *Boat* and *Train*, it can instead by stored in the node *Transportation*. In this way, we trade space against accuracy of cached information.

One challenge of the taxonomy compaction technique is to know where the compaction should be performed: which peer in the tree and which categories. We solve this problem by maintaining a counter for each peer in the taxonomy tree, which contains the number of categories (nodes) where the peer participates. When the size of the tree is over a certain threshold, compaction is performed for the peer that participates in most categories.

Other techniques that can be used include compacting the peer that has the largest amount of taxonomically similar categories, i.e., the categories have a high degree of similarity, for example being siblings instead of belonging to

distantly related categories. This technique gives better results in terms of reduced accuracy, but it is more expensive to perform.

After a certain amount of time the number of peers contributing to the taxonomy tree might become high. When this is the case, peers that are non-experts (i.e., having least number of resources within the category) are removed.

For large taxonomy trees the compaction can be expensive. In this case, it might be necessary to store taxonomy trees for each peer in addition to the taxonomy tree covering all known peers and categories.

Yet another issue is determining the point when the taxonomic information should be cached. There are at least two alternatives:

- Aggressive caching: simply cache all taxonomic results that are returned and do compaction when the tree gets too large.
- Selective caching: cache only what is deemed interesting/useful. This decision can for example be based on the number of resources a peer contains in the particular category, i.e., store the peer identifier for a node in the TCache only when the peer has many resources related to the particular category.

3.3. Soft cache-coherence

So far we have assumed that entries in the taxonomy cache have a lifetime limited by an associated expiry time and that they are removed after this time has expired. In some application areas this will be sufficient, and in very dynamic P2P systems it is probably difficult to improve on this scheme. However, in some P2P systems contents are relatively static and peers are stable. Examples of such P2P systems include super-peer architectures in general, and stable base stations/peers in mobile P2P systems like MobiShare [18] described above. In such systems, the (super-)peers can be assumed to be stable enough to be responsible for maintaining an approximation of cache coherence for the taxonomy entries cached on other nodes. This approximation of exact state we denote *soft cache coherence*. Thus, instead of (or in addition to) using TTL, a peer can store the entries until it either receives an invalidate or update request from the remote peer.[3]

In most cache coherence schemes, cache coherence is maintained by invalidate and update messages. An update requires that the cached data can be viewed as key/value tuples, i.e., object identifier and object contents. In our context, the identifier is the category that is cached and the value is the peer identifier. However, the cached categories are cached as a result of previous queries. This means that changes of categories stored at a remote peer is not of interest, except for the case when they disappear and then an invalidate message can be used. Thus update messages are not needed in our context.

In the case of soft cache coherence, a peer also needs to know about the peers having cached parts of its taxonomy.

An alternative to the soft cache coherence approach describe above, is to instead delete entries in the TCache, when it is discovered (at query time) that the peer supposed to contain resources within a particular category does not contain them as expected. This is discovered when a query is sent and the result set is empty (i.e., no result routed back containing the match for the category as it was supposed to happen).

3.4. Lazy distribution of taxonomy summaries

In addition to the distribution of taxonomy summaries resulting from query results, an additional technique that can be used is lazy distribution of taxonomy summaries, where peers proactively push some of the categories they store to the rest of the P2P system. This approach is similar to gossiping variants [5]. It is also possible to piggyback additional categories on the query results (this should obviously be a limited number).

Taxonomy categories in query results can also be stored at the nodes in the return path from the result peer to the query peer. This would also be stored in a taxonomy cache, improving subsequent routing.

4. Simulation and evaluation

In this section we first describe the simulation setup, and then report the results from the simulations.

4.1. Simulation setup

We have developed a simulation environment in JAVA for experimental evaluation. We use excerpts of the DMOZ taxonomy [4] for describing the resources on peers. In this set of experiments, we used the part of DMOZ taxonomy defined by "Arts/Movies/Titles/P". This particular taxonomy excerpt contains 358 categories. Each peer holds an instance of the taxonomy and classifies its resources to the taxonomy categories. We used synthetic network topologies that resemble random graphs, only the connectivity degree is constant and neighboring peers share 3-5 common neighbors, i.e., the network is more dense than a random graph. The results presented in this paper are based on a topology with 4000 peers, and with average connectivity degree of 8.

3 Note the interesting aspect that this cooperation can be considered a service: the peer subscribes to maintenance of the taxonomy caching, and will be notified upon changes.

4 Open Directory Project, http://dmoz.org/

For resource allocation to taxonomy categories, each resource is assigned to k taxonomy categories, with k a uniformly distributed random variable ($1 \leq k \leq MAX$), where MAX denotes the maximum number of categories that a resource may belong to. Note that these taxonomy categories are not necessarily leaf taxonomy nodes. Further, each resource is assigned to a taxonomy node, following a zipfian distribution with parameter $a_{res2tax}$ (in our experiments set to 1.2). We ensure that all taxonomy nodes contain at least one resource.

Regarding resource allocation to peers, the $80/20$ rule is adopted, which means that 20% of the peers hold 80% of the resources. The number of resources that each peer holds is generated by a uniform random distribution. The actual assignment is done as follows: first, the 80% of the peers are assigned resources randomly, then the 20% of the peers are assigned resources from nearby taxonomy categories (practically sibling and child nodes), in order to represent "experts" on some categories, which resembles the case in real world.

Each peer hosts a query generator that creates queries for taxonomy concepts. A random distribution for modeling queries can be tested. Most interesting is the case of zipfian query distribution, which is usually adopted to model user behavior. Queries consist of one taxonomy concept and may also carry a number indicating the number of wished results. We consider: a) queries for all (as many as possible) results, b) queries for the first n results.

Each peer can cache a limited number C_s of entries (taxonomy concepts or paths), which is a tuple:

$$t = (T_c, P, R_c, R_t, T_e)$$

that consists of a taxonomy concept T_c (or more generally a path) and a list of peer identifiers P, the number of resources R_c belonging to the concept, the total number of resources R_t and the entry's expiration time T_e. The "key" for a tuple is the combination of taxonomy concept T_c and peer identifier P. The cache replacement algorithm used is based on decreasing the expiration time of cache entries after each round of queries, and replacing the entry with the lowest expiration time. In this set of experiments, aggressive caching is employed.

We test and compare the performance of the following search strategies:

1. *Flooding:* Typical flooding search with a TTL parameter.
2. *Flooding with TCache:* Flooding with TTL, but employ jumps if there are cache hits. If there peer connectivity degree is d, then $d/2$ jumps are performed and $d/2$ neighboring peers are randomly selected to forward the query.

A number of peers N_{Qp} are randomly picked to act as querying peers, each of them producing N_q queries. The queries are processed sequentially, i.e., for each querying peer, send its first query, then its second query and so forth. Both strategies are evaluated using the same parameters (querying peers, queries, etc.), in order to have, as much as possible, comparable results. In our simulations we assume 20% of the peers to be querying peers.

4.2. Results and evaluation

We measure the following indices (average values):

- Number of contacted peers per query.
- Number of messages required to answer a query, which indicates the amount of information that has to be transferred.
- Recall, i.e., how much of the possible matches in the network are actually found.

We compare to the typical search strategy (flooding) against flooding with TCache. The results are summarized in Table 1.

The results show that the TCache significantly reduces the number of messages needed to process a query and the number of nodes that have to be contacted. This makes the approach particularly suitable for limited-bandwidth connections.

5. Conclusions and further work

Mobile devices can be connected in a P2P network making it easy to share resources with other participating users. An important challenge is to make it possible for users to find relevant contents stored at remote devices. In this paper, we have presented an approach to improve P2P search based on a variant of summary indexing, where 1) the contents of a peer are described by terms in a taxonomy, and 2) remote peers maintain a taxonomy overview instead of detailed indexing information. We have shown through simulations that our approach while achieving comparable recall compared to basic flooding significantly reduces 1) the number of messages needed for performing a query, and 2) the number of peers that have to be contacted during the query.

An important aspect of our approach is that not all peers have to use the same taxonomy. Taxonomy caching would be most beneficial if all used the same taxonomy. In order to increase quality even with many different taxonomies, ontologies could be used to integrate categories from different taxonomies. Further work will study this issue in more detail.

	N_C F	N_C T	N_M F	N_M T	Recall F	Recall T
TTL=1	7.8	6.7	7.8	7.0	0.0022	0.0019
TTL=3	45.3	53.4	166.7	166.0	0.0117	0.0149
TTL=5	110.6	158.0	524.7	523.9	0.0282	0.0717
TTL=7	199.9	346.8	1058.6	1057.7	0.0506	0.1835
TTL=9	305.6	583.1	1721.0	1719.6	0.0773	0.2930
TTL=11	437.7	840.3	2566.3	2566.0	0.1104	0.4012
TTL=13	586.7	1120.6	3536.5	3535.8	0.1477	0.4891
TTL=15	741.6	1372.4	4560.2	4558.7	0.1864	0.5755

Table 1. Simulation results. N_C denotes number of contacted peers per query, N_M number of messages required to answer a query. F denotes flooding and T denotes flooding using the TCache.

Acknowledgments

The authors would like to thank Jon Olav Hauglid for his help on improving the performance and scalability of the simulator used in our experiments.

References

[1] B. Bhattacharjee, S. Chawathe, V. Gopalakrishnan, P. Keleher, and B. Silaghi. Efficient Peer-to-Peer Searches Using Result-caching. In *Proceeding of the 2nd IPTPS*, 2003.

[2] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making gnutella-like P2P systems scalable. In *Proceedings of SIGCOMM '03*, 2003.

[3] A. Crespo and H. Garcia-Molina. Routing indices for peertopeer systems. In *Proceedings of ICDCS'2002*, 2002.

[4] A. Crespo and H. Garcia-Molina. Semantic Overlay Networks for P2P Systems. Technical report, Stanford University, 2002.

[5] F. M. Cuenca-Acuna, C. Peery, R. P. Martin, and T. D. Nguyen. PlanetP: using gossiping to build content addressable peer-to-peer information sharing communities. In *Proceedings of the 12th International Symposium on High-Performance Distributed Computing (HPDC-12 2003)*, 2003.

[6] C. Doulkeridis, K. Nørvåg, and M. Vazirgiannis. DESENT: decentralized and Distributed Semantic Overlay Generation in P2P Networks. Technical report, AUEB, 2005. Available from `http://www.db-net.aueb.gr/`.

[7] C. Doulkeridis, V. Zafeiris, and M. Vazirgiannis. The role of caching and context-awareness in P2P service discovery. In *Proceedings of MDM'2005*, 2005.

[8] C. Gkantsidis, M. Mihail, and A. Saberi. Random walks in peer-to-peer networks. In *Proceedings of INFOCOM'2004*, 2004.

[9] G. Koloniari and E. Pitoura. Content Based Routing of Path Queries in Peer-to-Peer Systems. In *Proceedings of EDBT'04*, 2004.

[10] A. Löser. Towards taxonomy based routing in P2P networks. In *Proceedings of The Second Workshop on Semantics in Peer-to-Peer and Grid Computing*, 2004.

[11] A. Löser, F. Naumann, W. Siberski, W. Nejdl, and U. Thaden. Semantic overlay clusters within super-peer networks. In *Proceedings of the 1st International Workshop on Databases, Informat ion Systems, and Peer-to-Peer Computing (DBISP2P)*, 2003.

[12] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmér, and T. Risch. EDUTELLA: a P2P networking infrastructure based on RDF. In *Proceedngs of WWW'2002*, 2002.

[13] S. Patro and Y. C. Hu. Transparent query caching in peer-to-peer overlay networks. In *Proceedings of IPDPS'2003*, 2003.

[14] L. Pireddu and M. Nascimento. Taxonomy-based routing indices for peer-to-peer networks. In *Proceedings of the SIGIR Workshop on Peer-to-Peer Information Retrieval*, 2004.

[15] L. Ramaswamy, B. Gedik, and L. Liu. Connectivity based Node Clustering in Decentralized Peer-to-Peer Networks. In *Proceedings of P2P'03*, 2003.

[16] J.-M. Saglio, M. Scholl, and T.-A. Ta. Efficient query processing in P2P networks of taxonomy based sources. In *Proceedings of CAiSE'2005*, 2005.

[17] Y. Tzitzikas and C. Meghini. Query evaluation in peer-to-peer networks of taxonomy-based sources. In *Proceedings of CoopIS/DOA/ODBASE'2003*, 2003.

[18] E. Valavanis, C. Ververidis, M. Vazirgiannis, G. C. Polyzos, and K. Nørvåg. MobiShare: sharing context-dependent data & services from mobile sources. In *Proceedings of the 2003 IEEE/WIC International Conference on Web Intelligence (WI 2003)*, 2003.

[19] S. Voulgaris, M. Jelasity, and M. van Steen. A robust and scalable peer-to-peer gossiping protocol. In *Proceedings of AP2PC'2003*, 2003.

[20] C. Wang, L. Xiao, Y. Liu, and P. Zheng. Distributed Caching and Adaptive Search in Multilayer P2P Networks. In *Proceedings of ICDCS'04*, 2004.

[21] L. Xu, C. Dai, W. Cai, S. Zhou, and A. Zhou. Towards adaptive probabilistic search in unstructured P2P systems. In *Proceedings of APWeb'2004*, 2004.

[22] D. Zeinalipour-Yazti, V. Kalogeraki, and D. Gunopulos. Exploiting locality for scalable information retrieval in peer-to-peer networks. *Information Systems*, 30(4):277–298, 2005.