DHTs over Peer Clusters for Distributed Information Retrieval

Odysseas Papapetrou Wolf Siberski Wolf-Tilo Balke Wolfgang Nejdl
L3S Research Center, Leibniz Universität Hannover, Germany
{papapetrou, siberski, balke, nejdl}@l3s.de

Abstract

Distributed Hash Tables (DHTs) are very efficient for querying based on key lookups, if only a small number of keys has to be registered by each individual peer. However, building huge term indexes, as required for IR-style keyword search, are impractical with plain DHTs. Due to the large sizes of document term vocabularies, joining peers cause huge amounts of key inserts, and subsequently large numbers of index maintenance messages. Thus, the key to exploiting DHTs for distributed information retrieval is to reduce index maintenance. We show that this can be achieved by combining DHTs with peer clustering. Peers are first clustered into communities, each of the communities having a representative super-peer. Then all occurrences of a term in a community are published to the global DHT in a batch by the representative super-peer. Our evaluation shows that this reduces index maintenance cost by an order of magnitude, while still keeping a complete and correct term index for query processing.

1 Introduction

The introduction of distributed hash tables (DHTs) for the organization and indexing of data in peer-to-peer networks has led to a boost in scalability, robustness, and efficiency [23, 21]. On the one hand, in contrast to simple flooding approaches all network content is visible from every peer. On the other hand, DHT algorithms efficiently distribute the lookup process over all peers. Moreover, effective schemes for load balancing [9] have also addressed problems with imbalances introduced, e.g., by Zipf-distributed query load or by different capabilities of peers.

However, using DHTs for fulltext indexing poses severe problems. Even after traditional preprocessing steps like removing stopwords and stemming, documents still feature a large amount of different keywords that need to be indexed. All new content of a joining or updated peer has to be published in the DHT. In fact, since in DHTs a hashing function decides which peer is responsible for which terms, usually a considerable number of peers holding some part of the DHT have to be contacted to fully publish all the information about each peer.

In this paper we present a novel approach to realize efficient peer-to-peer document retrieval, by combining a super-peer architecture with a DHT. We organize peers into groups, each of them being represented by a super-peer for publishing the group's information to a single global DHT used for query processing. Each of the peers independently joins a group (either randomly, or based on its collection), and submits its index to the representative super peer for the cluster. The group representatives efficiently batch this information and periodically forward aggregated publishing/update messages to the DHT. Hence the number of the required DHT lookups for publishing and the total number of messages is significantly reduced. Indeed, our evaluation shows that the proposed approach enables message savings of as much as one order of magnitude compared to a plain DHT, regarding number of messages as well as total transfer volume. Thus, the process of maintaining the distributed keyword index is significantly less expensive compared to traditional DHTs, making complete term indexing feasible.

While peer grouping alone already offers significant performance improvements, we can achieve further gains by introducing a distributed clustering scheme, such that peers in each individual group have similar contents. This reduces the total number of the words needed to index in each cluster and additionally decreases DHT lookups needed for publishing and the total number of messages for maintaining the DHT.

The paper is structured as follows. After reviewing the related work on the area, we present the basic algorithm and building blocks of our topology in Section 3. We then introduce our distributed clustering scheme, targeting to even less network overhead for maintaining the index. Section 5 contains the results of our experimental evaluation. We conclude with a discussion and future work.

2 Related Work

The problem of peer-to-peer indexing of large document collections has been recognized already early. The area of peer-to-peer information retrieval has first focused on distributing retrieval in unstructured networks, using approximated collection-wide information. PlanetP [6] is one of the first such systems. It uses gossiping to distribute peer content summaries, represented by bloom filters. From these summaries each peer computes inverted peer frequencies, which are used to rank sources for a given query in a similar fashion as in GlOSS [10], but now without a central coordinating instance. Due to the gossiping of collection-wide information, such approaches exhibit only limited scalability.

Since DHT infrastructures offer a lot of efficiency and scalability over unstructured networks, the idea of opening up DHTs also for IR-style retrieval is appealing. However, Loo et al. [12] have shown that a pure DHT solution does not scale for fulltext indexing, because index maintenance becomes too expensive. ALVIS [20] increases DHT scalability by indexing per peer only those keywords most significant for describing each document in its collection, thus trading the index' completeness for improved scalability. As it is still a difficult problem to decide locally which terms should be considered most significant with respect to the global collection, [22] proposes a similar approach, PNear, where the terms to be indexed are chosen randomly from the peers' abstracts. Even with this simplification already good performance benefits can be obtained, but again at the price of index incompleteness. The Adlib approach [8] establishes a two-tier structure, where a first tier divides the documents into independent, equal-sized partitions, so-called domains. Within each domain, nodes build a distributed index over the content stored which then is offered in the second tier for querying. By tuning the size and number of domains index maintenance can be traded-off against query efficiency.

Recently the concept of hierarchical DHTs has been introduced to foster efficient bandwidth utilization and a better adaptation to the underlying physical network, see e.g., [7, 24]. Especially the latter point is most promising for employing hierarchical DHTs also for information retrieval tasks in peer-to-peer networks. However, the current proposals for hierarchical DHTs are still less effective in messages compared to flat DHTs [24].

Super-peer-based algorithms have also been proposed in this area. [5] proposes to maintain collection-wide information as well as routing tables at designated super-peers, called infobeacons. [2] collect query statistics at super-peers to facilitate efficient keyword query processing. However, querying does still not scale as good as with DHTs (i.e., not logarithmically). To our knowledge, the combination of DHT and super-peers for P2P information retrieval has not been proposed yet.

3 Basic Algorithm

Let us now present our basic algorithm. Figure 1 illustrates our approach: all peers first join a global Distributed Hash Table, however, without publishing any of their contents. Then each of them autonomously joins a group by attaching to a suitable super-peer, which takes over the responsibility of publishing the entire group's keywords to the DHT and keeping it updated. Each peer updates only the super-peer of its group with the inverted term index of its collection. In turn, the super-peers periodically update the DHT by publishing the inverted indices of their peers' collections. Please note that the super-peers indeed post the original indices as received from their group peers, not only an aggregated inverted index for the group. Thus, a complete index is still maintained at the DHT, and the query routing process does not involve the super-peers (cf. 3.2).

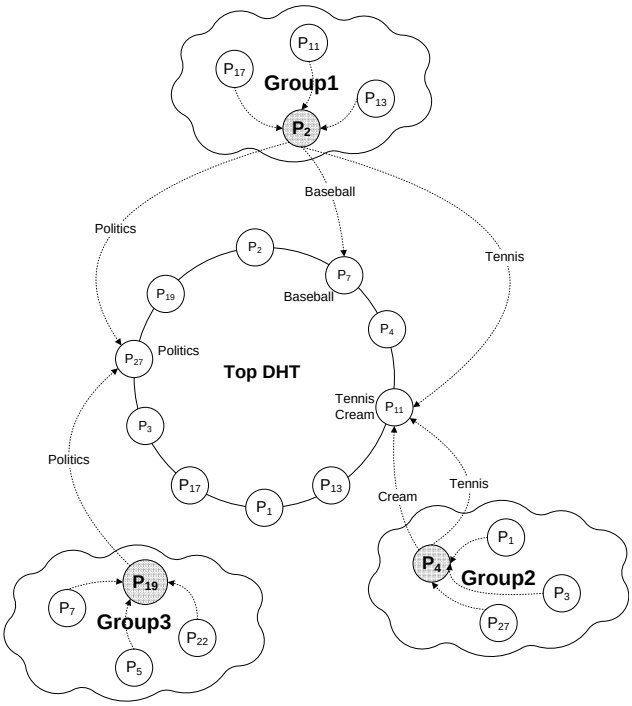

Figure 1. Network Overview:Super-peers are gray-shaded

Peer inverted index: For full-text indexing, peers locally extract the inverted index for their collection: for each term, the occurrences in the peer are counted (peer frequency), after all terms have been stemmed and stopwords have been

removed as usual. The maximum peer frequency (the frequency of the most frequent term in each peer) can also be determined for assessing how discriminating each term is with respect to the local collection. Both peer and maximum peer frequency are needed for the collection selection strategy during query evaluation. Unlike document-granularity meta data, meta-data on peer-granularity level is compact and efficient to exchange in a distributed system, especially for full-text indexing scenarios. While it may result in less precision in query execution, it provides a good balance between precision and resource requirements [4].

DHT index: The inverted index of relevant peers for each term is built using a Distributed Hash Table, including index information for *all* peers in the network at any given moment, not only the super-peers. New terms to be published are automatically mapped to peers by the DHT using a hash function. For each term, the responsible peer in the DHT then keeps a list of all relevant peers and their peer scores: the peer frequency (PF), and the maximum peer frequency for each peer (Max PF)(see table 1). In Section 3.1 we explain in detail how this table is built and maintained; Section 3.2 shows how this information is used to evaluate queries.

Keyword	Peer:IP Address	PF	Max PF	Super-peer: IP Address
Tennis	$P_{11} : IP_{11}$	14	43	$P_2 : IP_2$
	$P_1 : IP_1$	35	57	$P_4 : IP_4$
	$P_3 : IP_3$	32	64	$P_4 : IP_4$
Cream	$P_3 : IP_3$	34	64	$P_4 : IP_4$
...	...	...	...	...

Table 1. Logical Top DHT Routing Table

In our approach, any DHT implementation can be used, since it only requires the generic functionality of all DHT overlays. For our evaluation, we used Chord [23] as DHT infrastructure. However, in the algorithm description we just refer to the generic DHT insert/lookup functionality and abstract from implementation details .

3.1 Peers Lifecycle

Let us now take a closer look at the lifecycle of the peers in the network.

Initialization: The first task of any new peer is to join the DHT, which is facilitated by using the DHT's protocol. The joining peer then needs to decide whether it should serve as super-peer or just join as normal peer. Unlike previous super-peer systems (e.g., [17]), our system does not require the super-peers to be especially powerful or have a very fast connection. A long up-time of super-peers is also not required, but reduces the number of maintenance messages needed. As the size of its peer group can be individually set by each super-peer, the additional workload of super-peers is adaptable.

In summary, a joining peer can either become a super-peer and create a new group, or join an existing group. The decision for this can be made by each peer independently, for instance based on the desired ratio of peers and super-peers, which in turn determines the probability of each individual peer to serve as super-peer P_{SP}. Each joining peer then draws a random number and can decide based on P_{SP} whether it becomes a super-peer or a normal peer. In the first case, it publishes its contents to the DHT directly, and awaits other peers to join its peer group. In the second case, the peer needs to find a super-peer that still accepts more connections. Our approach implements this by the peer running a random query on the DHT, in order to discover some super-peers (in the case of Chord, which typically has an id ring from 0 to 2^{160}, the random query can for example be a random number on this range). The query will return a peer responsible for holding the respective hash value, which will be in turn queried for all the super-peers it is aware of (i.e. that have published information in its local term frequency inverted list). The retrieved super-peers are then checked in random order. The new peer joins the peer group of the first discovered super-peer that accepts it.

Peer publishing: After a peer has joined a group, it sends its inverted index to the super-peer of the group. The advantages as compared to publishing all inverted indices directly to DHT are:

1. All the peers are directly connected to the super-peer of their group. They thus do not have to query the DHT keyword by keyword for publishing their inverted index, which in the case of n peers would require $O(\log(n))$ messages for each distinct term at each peer.
2. In contrast to publishing each individual term in a DHT, the inverted index can be sent to the super-peer in a single message and thus can be efficiently compressed.

In our approach, we rely on periodic sending of the peer indices to the super-peers. The super-peers use a sliding-window technique to maintain the information for their respective groups. Thus, the only requirement for the normal peers is that the period for re-sending their information is (at most) equal to the length of this sliding window. A delta-updating, or a single *PING* message to notify a super-peer that some peer is still alive, can also be used to significantly reduce the number and size of the messages. Note

that optimizations like the delta-updating or the *PING* message are not possible in the individual DHT publishing scenario, since they would also have to be sent to all affected DHT peers, and thus not save messages.

Super-peer publishing: The inverted index as published in the global DHT is based on periodic re-publishing by all super-peers. The super-peers group peer frequencies per term, and publish them in the DHT. Meta-data for all the group's peers with respect to each term is grouped in a single message, thus (a) requiring only a single DHT lookup and only one publishing message per term, and (b) enabling efficient compression as well as delta-updating. A small disadvantage with respect to the message grouping is that although less messages will be needed, each individual message may get bigger. This is because messages now comprise all information for a term with respect to all the peers in the group. This slight overhead, however, is easily outperformed by the significantly fewer required DHT lookups, each of which costing $\log(n)$ messages.

Peers leaving the network: Since we rely on periodic re-publishing, we do not have to act in the case of a normal peer leaving the network (either by expected departure or by unexpected failure): the super-peers and the global DHT itself will automatically recover without loss of data (see for example [23, 1]). The normal peer's summary published at the super-peer will also eventually expire, get removed from the super-peer and in turn from the DHT inverted index. Any query routed to this peer in the meantime will simply fail, and the next promising peer will be selected for routing.

Whenever a super-peer leaves the network, all connected peers in its group need to re-attach themselves to a different group, or create their own group by becoming super-peers themselves. If the disconnection of the super-peer is announced, the super-peer transfers responsibility for the group to any of its connected peers before departure. It transmits the group's collected information to the new super-peer (such that the group's peers do not need to re-build it from scratch but can keep updating it), and notifies all the peers in its group about the change. In case of an unexpected failure of a super-peer the peers of its group individually reconnect to the network and join another group as described above. In either case, the new super-peer does not need to re-publish the information to the global DHT immediately, so no extra publishing cost occurs from changing the super-peer. Instead, the group's information needs to be re-published, refreshed or updated only prior the expiration of the old information.

3.2 Query Processing

Query Processing in the basic approach is a two step process: (a) first all relevant peers need to be detected (the collection selection problem), and (b) the top α relevant peers have to be queried to return the results to the query initiator. Note that during the query processing task the super-peers behave just like normal peers: they are not required to act as a single entry point for their group. Instead, direct communication with all peers is facilitated using the standard query features of the specific DHT implementation.

Peer selection: There is rich literature on collection selection techniques in distributed search. GlOSS [10] creates a centralized meta-data repository which enables collection ranking for queries. CORI [4] represents collections as huge documents, and modifies the $TF*IDF$ function accordingly. DTF [18] proposes a theoretic framework which minimizes the time and cost and maximizes the retrieval quality of the query.

The distributed indexing model we provide in our approach enables the use of such standard collection selection algorithms. In fact, we use CORI for peer selection, due to its performance and simplicity. CORI performs well and gracefully scales for large networks. The CORI score s_i for each collection i for each query q is calculated with the following formula:

$$s_i = \sum_{t \in q} \frac{d_b + (1 - d_b) * T_{i,t} * I_{i,t}}{|q|}$$

with $T_{i,t} = d_t + (1 - d_t) * \frac{\log(df_{i,t}+0.5)}{\log(max_{df_i}+1.0)}$ and $I_{i,t} = \frac{\log(\frac{|C|+0.5}{cf_t})}{\log(|C|+1.0)}$.

d_t and d_b are constants with a default value of 0.4 (for more information on how to set these constants see [4]). $df_{i,t}$ is the document frequency of term t in collection i (we refer to it as Peer Frequency in our DHT setup) and cf_t is the collection frequency, that is, the number of collections where t occurs at least once. max_{df_i} is the maximum document frequency of the most frequent term in collection i (what we call Max. Peer Frequency at the DHT). $|C|$ is the number of collections (in our case an estimation on the total number of peers).

Comparing the data stored in the DHT (Table 1) and the data required from the CORI algorithm, we can see that all required data is already stored in the DHT, except for the estimation of the total number of collections $|C|$. Thus, for each query term the necessary parameters can be easily retrieved by only one DHT lookup per term. The number of collections (peers) $|C|$ is a rather stable attribute of the DHT (practically required from most of the query routing algorithms), and can be inexpensively estimated over a DHT [13, 11]. In our system we use an efficient proposal based on random walk [13] to estimate $|C|$.

Query execution: After the top-α most promising peers have been identified, the query is routed to each individual peer for execution. The respective Inverse Collection Frequency *ICF* (the number of the peers that carry the term at least once) for each query term is attached to the query, to allow all peers to properly weight the importance of each term, and select the top-k documents, using a normal $TF*IDF$ score. Links to the results are returned to the query initiator, ordered by relevance with respect to the query.

Since it simply relies on the base functions of the DHT, the query routing technique in our approach is not a novel contribution of this work. We only selected it as one out of the many possible query execution techniques; for implementation details please refer to the original work cited above. For the same reason it is also out of the scope of this work to evaluate the precision-recall or the performance/execution complexity of query execution: they are exactly the same as in the cited papers. In contrast, our focus is on efficiently maintaining the required information for arbitrary query execution techniques that work on DHTs.

4 Clustering-enhanced Algorithm

Though the basic algorithm has a slight overhead by first sending index information to the super-peers and then publishing it in the DHT, we will show in the evaluation that it still outperforms the naive publishing of all vocabulary terms by each peer by far. This gain is due to the efficient handling made possible by overlapping terms in the super-peers, which requires significantly less DHT lookup messages.

In fact, the communication between the peers and their super-peers is inexpensive: no DHT lookups are needed, the messages can be efficiently compressed, and delta-publishing can be used to further reduce the size of the messages. In contrast, communication between the super-peers and the DHT overlay for actually publishing the group's data is the expensive part of the publishing process: it requires many DHT lookups, each of them generating $log(n)$ messages. Thus, reducing the number of DHT lookups that the super-peers require can essentially reduce the DHT maintenance cost.

In our basic algorithm peers just group randomly around a super-peer. In this section we explore how DHT maintenance costs can be even further reduced by a more sophisticated peer grouping/clustering based on their respective vocabulary. Since such a clustering increases the term overlap between the peers in the cluster, it decreases the number of distinct terms within each cluster. As a result, the super-peer responsible for that cluster will need to perform fewer DHT lookups for publishing the total terms of the collections.

While decreasing the distinct terms per super-peer by increasing the expected overlap between terms of the peers in the same group, we keep the average capacity of the super-peers as in the basic approach, but replace the randomly-created groups of the basic algorithm with clusters of similar peers. For this, we need an efficient distributed clustering algorithm to assign peers with similar terms to the same cluster. The algorithm touches two parts of the basic random-grouping algorithm:(a) the super-peer publishing in the DHT now also includes aggregated cluster information which enable efficient selection of "good" clusters for joining peers, and (b) the process of joining peers which is now performed in an overlap-increasing approach. In all other respects the clustering enhanced algorithm just works like the basic algorithm.

4.1 Aggregated Cluster Information Publishing

We slightly enhance the DHT structure to enable efficient overlap-increasing cluster selection for joining peers. Apart from the virtual peer granularity information, each cluster's super-peer now publishes in addition aggregated cluster information for its cluster. For each of the cluster's top-λ most frequent words, the super-peer publishes an extra record to the DHT, including the overall cluster frequency. The cluster-granularity records can be easily distinguished from the peer-granularity records, since the *Peer:IP* field in the former is empty (see, for example, Table 2).

Keyword	Peer:IP Address	PF	Max PF	Super-peer: IP Address
Tennis	$P_{11} : IP_{11}$	14	43	$P_2 : IP_2$
	$P_1 : IP_1$	35	57	$P_4 : IP_4$
	$P_3 : IP_3$	32	64	$P_4 : IP_4$
	null	67	78	$P_4 : IP_4$
Cream	$P_3 : IP_3$	34	64	$P_4 : IP_4$
...	...	...	...	...

Table 2. Logical Top DHT Routing Table for clustering-enhanced approach

Note that the additional cluster-granularity data is of negligible size and requires no extra messages. A typical value for the top-λ keyword is lower than 10, which generates a total of 10 additional bytes. Furthermore, the additional data can be piggybacked on the existing DHT-publishing messages.

4.2 Selecting Clusters with Maximum Overlap

The main difference between the clustering-enhanced approach and the basic algorithm is on how the peers select super-peers when they join the network: now peers require support for a fully distributed clustering mechanism. The clustering mechanism is supported by an enhancement in the DHT layer, such that: (a) the cluster centroids can be easily retrieved by new peers, and (b) the new peers can efficiently find clusters similar to their collection, without comparison to all the different centroids.

Apart from the aggregated cluster information publishing (section 4.1) we also need a compact representation for the cluster centroids. In literature, cluster centroids are usually represented by variations of a term frequency matrix keeping some statistics for each of the terms occurring in the cluster. However, this is very expensive for collections in distributed scenarios, since centroids need to be transfered over the network. Since our approach only aims at reducing the number of distinct terms within a cluster, the term frequency can be safely ignored. Hence, we use simple bloom filters for representing cluster centroids and peer collections.

Computing and representing cluster centroids: The super-peer of each cluster is responsible to create the cluster centroid. To do that super-peers already have all the required information from their group's peers, i.e. the keywords from their inverted indices. The super-peer now uses a bloom filter to to represent the cluster centroid. The Bloom filter data structure was first proposed in [3], as a space-efficient representation of sets $S = \{e_1, e_2, e_3 \dots e_n\}$ of n elements from a universe U. A Bloom filter consists of an array of m bits and a set of k independent hash functions $F = \{f_1, f_2 \dots f_k\}$, which hash elements of U to an integer in the range of $[1, m]$. The m bits are initially set to 0 in an empty bloom filter[1]. An element e is inserted into the bloom filter by setting all positions $f_i(e)$ of the bit array to 1.

Bloom filters allow efficient membership queries: for any given element $e \in U$, we can safely conclude that e is not present in the original collection if at least one of the positions computed by the hash functions of the bloom filter points to a bit which is set to 0. However, Bloom filters suffer from false positives; due to hash collisions, it is possible that all bits representing a certain element have been set to 1 by the insertion of other elements. The probability that such a membership test yields a false positive is $P(false-positive) \approx (1-e^{-kn/m})^k$. The information density of a bit filter is optimal when the probability of each bit to be set is $1/2$. For a bloom filter, this is the case when setting the number of hash functions to $k \approx \frac{m}{n} * \ln(2)$.

[1]We use the expressions 'A bit is set to true/false' and 'A bit is set to 1/0' interchangeably.

Clustering objective function: The objective function for our clustering counts the number of the bloom filter bits that have to be changed from false to true in the cluster summary, if the peer joins this cluster. Note that each cluster has an upper limit for peers, so the clusters' bloom filters are not expected to become too dense, which would constantly result in a low number for the objective function. Formally the objective function for choosing a cluster for a peer can be defined as:

Definition Let the new peer be P_i, and the bloom filter for its collection be BF_{P_i}. For all candidate clusters C_x, with bloom filter centroids BF_{C_x}, the objective function $f(P_i, C_x)$ is:

$$f(P_i, C_x) = diff((BF_{C_x}\ or\ BF_{P_i}), BF_{C_x}) \quad (1)$$

where $diff(BF_i, BF_j)$ equals the number of bits that the two bloom filters differ in. The best cluster is the one that *minimizes* the objective function.

The above definition indeed leads to the desired clustering. It can be shown that with high probability the number of new words added on a collection reduces when the objective function gives lower results. The proof is based on an operation for estimating the number of elements hashed to a Bloom filter [16]. For brevity we only sketch the proof here. The full proof is found in [19] :

1. Let the Bloom filters BF_{C_1} and BF_{C_2} represent the centroids of two collections C_1 and C_2, respectively. Furthermore, let the Bloom filter of some new peer p be denoted as BF_p and assume that all Bloom filters are created using the same hash functions.

2. Then the total number of elements n_1 and n_2 hashed in BF_1 and BF_2, respectively, can be estimated with some confidence level using the operation provided by [16].

3. Now assume that adding p's collection to C_1 results in z_1 bits to be switched from 0 to 1 in BF_{C_1}, analogously z_2 for BF_{C_2}. Estimating the number of elements in the changed BF_{C_1} gives us a new expected n'_1. We calculate n'_2 analogously for C_2. Of course for calculating n'_1 and n'_2 we have to use the same confidence level as in step 2.

4. Finally $n'_1 - n_1$ gives the expected number of *new* terms that will be added in C_1 and grows with z_1 (likewise for C_2 and z_2). Selecting the cluster that minimizes our objective function therefore means that less new terms will be added to the cluster, and many terms will overlap with the existing cluster terms.

In summary, the proposed cluster centroids are simple to build, without any extra network overhead. The approach creates acceptable clusters and significantly reduces the distinct terms per cluster as compared to the random grouping method of our basic algorithm.

4.3 Managing Collection Diversity by Virtual Peers

Real-life peer collections, as real persons' interests, are often quite diverse with respect to the topics of interest: a peer may collect documents about the topic of *jazz music*, and at the same time documents on *spontaneous nuclear fission* and *football results*. Trying to find the best cluster for a multi-thematic peer is difficult, and may lead to suboptimal clustering.

An approach for solving this problem is to split each peer into a set of virtual peers with a more homogeneous topics each by document clustering. Ideally, each virtual peer should be focusing on only a single subject, so that efficient clustering around super-peers can be performed. Then, each virtual peer can act independently, join the best-matching cluster, and post its contents to the super-peer, that in turn will have a more homogeneous collections of terms to post to the global DHT.

The partitioning of each peer's collections into virtual peers is a local process. A thematic clustering is performed using only the local information present in the peer's collection. Each peer is free to select a suitable number of clusters and the actual clustering algorithm for determining its virtual peers. The peers can even choose to use a partitioning hierarchy e.g. the file system structure or an ontology like MeSH for medical documents, for creating suitable clusters instead of employing a statistical clustering mechanism.

4.4 Joining Peers in the Clustering-Enhanced Algorithm

Now we are ready to present how the actual joining of peers is handled by the cluster-enhanced algorithm. Like in the basic algorithm the first step of a new peer joining the network is to join the DHT; again all peers contribute resources to the DHT. In a second step, each new peer splits itself to a set of virtual peers, using local document clustering as explained in section 4.3. Each of the virtual peers then independently finds a suitable cluster in the P2P network. If no suitable cluster is found, or if a peer would introduce to many new terms into all clusters, it creates a new cluster and becomes the respective super-peer. Searching for a suitable cluster is based on normal DHT lookups:

1. The virtual peer computes the term frequency list for its collection, which it uses to detect the λ most frequent terms, $T_1, T_2, \ldots T_\lambda$
2. For all terms $T_i \in T_1, T_2, \ldots T_\lambda$, the virtual peer performs a DHT lookup, and retrieves the information associated to all the clusters for which T_i is in the top-λ terms of the cluster. Note that each cluster publishes the score for its top-λ terms in the DHT (see table 2). Only one DHT lookup is required for each term and the individual super-peers need not be contacted yet.
3. If no candidate super-peer is found (i.e. none of the terms $T_1, T_2, \ldots T_\lambda$ belongs in the top-λ keywords of the existing super-peers), the virtual peer creates a new cluster, and becomes a super-peer responsible for that cluster.
4. If candidate super-peers are found, they are ordered based on their aggregated relevance to all the λ terms (i.e. using a TF*IDF score). The top-μ super-peers based on this order are extracted.
5. The virtual peer sends its centroid/bloom filter at the top μ super-peers asking for the distance between the cluster centroids and its own centroid. The super-peers compute the distance using the objective function (Equation 1) and return it at the virtual peer. The choice of putting the burden of distance computation at the super-peers instead of the joining peers was made because the joining peers' centroids can easier be transferred over the network compared to the cluster centroids (due to lower sparsity, enabling higher bloom filter compression).
6. Finally the virtual peer joins the optimal cluster, according to the objective function results.

The above procedure incurs max. $\lambda * \log(n) + \mu$ total messages. After a virtual peer joins a cluster, it periodically sends its inverted index to the super-peer of the cluster. Again this requires only one message per virtual peer, and can be effectively optimized by compression and delta-submission. When a virtual peer sends its information, the respective super-peer updates the cluster centroid to reflect all current information.

Peers leaving the network (expected or unexpected) and the actual querying are handled exactly in the same way as described in Section 3.

5 Evaluation

We evaluated our two proposed approaches using real document collections from MEDLINE [14], and compared to the traditional DHT publish/subscribe approach. The MEDLINE data set is a database produced by the US National Library of Medicine, and currently contains information for more than 11 million citations. Most of the

citations are accompanied with abstracts, and are annotated according to the MeSH (Medical Subject Headings) database [15]. MeSH includes a controlled set of headings of various specificities, for annotating the citations.

Our document collection was created as follows: 200 headings were randomly selected from the MeSH hierarchy. For each heading, 800 citations annotated with this heading were selected. The citations included title and abstract. This resulted in a total of 160.000 documents. This document subset was used to build the peer collections.

Building the peer collections: Real-life peer collections, as in real persons' interests, are often multi-thematic. Some users may be well-focused, having very specific documents of only one topic. Other users may focus on a couple of non-related topics, while still others may just collect many diverse documents. We take this into account by assigning collections from several MeSH categories to each peer.

The MeSH headings were also used to split each peer to virtual peers in the clustering-enhanced approach. While we cannot expect a MeSH-like annotation for all real P2P collections, MeSH headings were only used as inexpensive means to implement local clustering, and can be replaced with any text clustering algorithm. In fact, we expect much better performance from a normal clustering algorithm which uses the actual overlap between document terms, since that will actually lead to clusters with fewer distinct terms.

Evaluation Results: We evaluated our algorithms with six different network sizes: 500, 1000, 2000, 3000, 4000 and 5000 peers. Each peer randomly selected 3 MeSH headings out of the 200, and, for each of the headings, 20 random documents out of the 800 available. The peer collections for each network size were rebuilt three times, and the experiment was repeated 4 times for each peer collection, yielding a total of 12 different setups.

In the experiments we compared index maintenance costs of the basic and clustering-enhanced setups with the requirements of the flat DHT approach, where each peer independently publishes the frequency for each term at its collection at the DHT. We measured number of messages as well as the total data transfer volume.

Number of messages: All messages were counted, including the DHT-lookup messages. Note that each DHT lookup is generating approx. $log(n)$ distinct DHT lookup messages.

Transfer volume: We measured the total sum of message sizes. Apart from the data included in the messages, we included a network header overhead of 40 bytes for each message (theoretical minimum for network message). Furthermore, GZIP message body compression was used wherever this was reducing the size of the message. It turned out that this is the case only when submitting the peer index from the virtual peer to the super-peer, or when exchanging bloom filters.

All the approaches relied on periodic republishing: the full peer data was submitted periodically. We did not include data churn in the above experiments, since under the periodic republishing model, the churn does not have an effect in the messages. The basic and the clustering-enhanced setups were configured as follows:

Centroid bloom filter length: The clustering-enhanced approach was executed with bloom filter lengths of 16, 32, 64 and 128 Kbits, to see which size yields optimal results.

Maximum peers per cluster: The maximum peers per cluster for the clustering-enhanced approach were determined based on the density of cluster's centroid (bloom filter). A maximum density of 50% was allowed. After that, the cluster was not allowed to get more peers.

Total number of clusters: To get a fair comparison, we made sure that both the basic and the clustering enhanced algorithm have the same number of clusters, so that the super-peers have the same workload on average. Since the number of clusters in the clustering-enhanced approach is dynamically determined (see joining peers algorithm at section 4.4), the clustering-enhanced approach was first executed, and the same number of random clusters were then created for the basic algorithm for comparison.

Top-λ keywords: A value of 4 for λ was found to be a good trade-off between clustering quality and cost in our preliminary experiments.

Top-μ super-peers: A value of 3 for μ was found to be sufficient in our experiments.

The results clearly show the benefits of our approaches. Figure 2 plots the number of messages (left-hand scale) and the total transfer volume for different network sizes. All results are normalized on the network size (i.e., divided by the number of peers), and the plot is tagged with the total number of created clusters (average over all the runs) for each setup. In all experiments, the basic and clustering-enhanced algorithm easily outperform the flat DHT model. The performance gain of the two proposed methods compared to the flat DHT model increases with network size. Furthermore, the clustering-enhanced algorithm was better than the basic algorithm: the cost for finding the most similar cluster is lower than the benefit of higher keyword overlap at the super-peers. Due to lack of space, we only present results

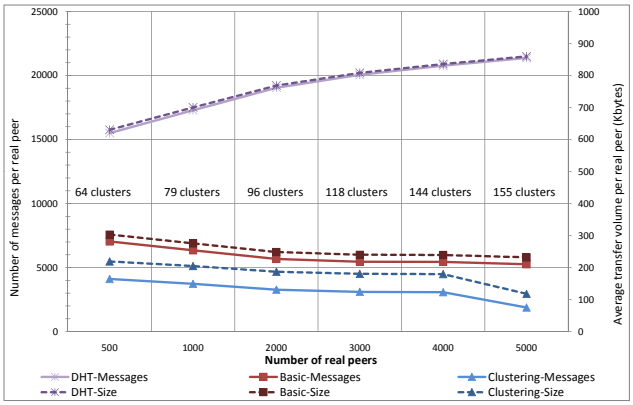

Figure 2. Messaging requirements for different network sizes

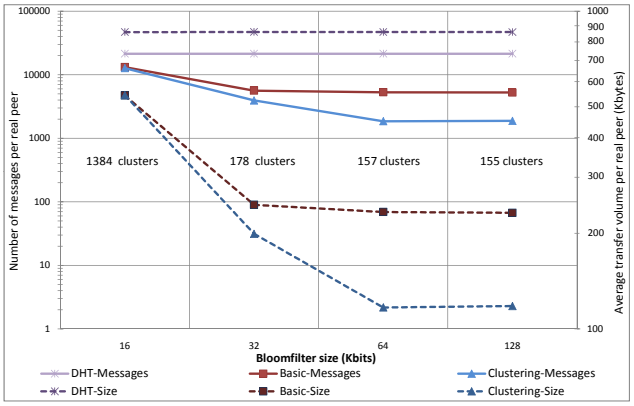

Figure 3. Effect of Bloom filter size for a fixed network of 5000 peers

for bloom filter length equal to 128Kbits. For the full results the reader is referred to the long version of the paper [19].

Figure 3 presents the effect of the cluster centroid's bloom filter size, for the 5000 peers setup. The clustering approach is plotted with the flat DHT approach and the basic algorithm for comparison purposes. Note that we use logarithmic scales in this figure. A bloom filter size of 16 Kbits quickly becomes more than 50% dense, causing the cluster not to accept new virtual peers. This significantly reduces the average cluster size and does not allow efficient clustering or significant keyword overlap. Too many very small clusters are created. The extra cost of each peer to find the *most similar* cluster neutralizes the advantage of the small keyword overlap in the cluster. Thus the messaging requirements between the basic and the clustering-enhanced versions are practically the same. A bloom filter size of 64 Kbits seems to be better suited for clustering. At the opposite extreme, 128 Kbits for this setup appear to be not necessary, incurring a slightly higher cost than the 64 Kbits centroids case.

6 Conclusions and Outlook

In this paper we presented a hybrid DHT/super peer P2P system that significantly reduces the network cost for maintaining a full inverted index within a global DHT. Because individually publishing all terms of each peers vocabulary to a global DHT is far too expensive, we group peers around super-peers that are responsible for integrating all information about their connected peers' collections, and publish it to the global DHT. The message gain is thus achieved by reducing expensive DHT lookups for a large number of terms. Experiments with practical data sets indeed show a massive gain of about an order of magnitude over the conventional periodic publish/subscribe approach in a global DHT.

We presented two algorithms differing in the way peers actually group around super-peers: the basic algorithm groups peers randomly around super-peers, whereas the cluster-enhanced approach first splits peers into thematically focused virtual peers that can then be clustered more effectively, to foster group homogeneity. Whether creating random groups of peers or using clustering algorithms for grouping, a sizable term overlap within each group can be observed. This overlap allows the responsible super-peer to reduce the number of DHT inserts for the actual publishing of all the terms. Our clustering algorithm increases the term overlap in each group significantly, and thus requires even less DHT lookups than the basic approach. Both approaches reduce the network cost without having negative effects on the querying precision/recall and performance of the original DHT model. To the best of our knowledge, this is the first work proposing a hybrid DHT/super-peer model, and using this model to derive an essential message gain.

Our future work will address mainly two topics implementing more sophisticated information retrieval techniques, and better peer clustering.

Information retrieval techniques: While our work gives already significantly better costs for the inverted index maintenance, we still did not reach the full capabilities of the information retrieval model. In particular, we did not yet address the limitations of the simple $TF * IDF$ retrieval model. For instance, currently conjunctive queries are handled by intersecting lists from different peers in the DHT. Using intra-cluster communication, peers in a cluster can efficiently realize information retrieval techniques beyond the inverted index, which are too complex to run with all the peers in the flat DHT. Consider for instance a two-step query execution technique: (a) first find a proper cluster, and (b) then find the right peers to query within the clus-

ter. Since the number of peers per cluster is rather limited (as opposed to the total number of peers in the network), even complex information retrieval infrastructures can be employed for query evaluation within each cluster. At the same time, we can limit the inverted index over the global DHT to aggregated data for the clusters.

Better Peer Clustering: We expect that a more semantically motivated grouping of peers to clusters will increase term overlapping even more and further reduce the distinct terms per cluster. Hence, we will evaluate more complex distributed clustering algorithms and study their execution cost in our DHT scenario. This includes the optimization of cluster number size, as well as when and how to merge/break clusters.

References

[1] K. Aberer, P. Cudré-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Punceva, and R. Schmidt. P-Grid: a self-organizing structured P2P system. *SIGMOD Rec.*, 32(3):29–33, 2003.

[2] W.-T. Balke, W. Nejdl, W. Siberski, and U. Thaden. Progressive distributed top-k retrieval in peer-to-peer networks. In *Proceeding of Proceedings of the 21st International Conference on Data Engineering (ICDE)*, Tokyo, Japan, 2005.

[3] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications ACM*, 13(7):422–426, 1970.

[4] J. P. Callan, Z. Lu, and W. B. Croft. Searching Distributed Collections with Inference Networks . In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–28, Seattle, Washington, 1995. ACM Press.

[5] B. Cooper. Guiding queries to information sources with InfoBeacons. In *ACM/IFIP/USENIX 5th International Middleware Conference*. ACM, 2004.

[6] 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 *Twelfth IEEE International Symposium on High Performance Distributed Computing (HPDC-12)*. IEEE Press, June 2003.

[7] P. Ganesan, P. K. Gummadi, and H. Garcia-Molina. Canon in G major: Designing DHTs with hierarchical structure. In *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS)*, pages 263–272, Tokyo, Japan, 2004.

[8] P. Ganesan, Q. Sun, and H. Garcia-Molina. Adlib: A self-tuning index for dynamic peer-to-peer systems. In *Proceedings of the 21st International Conference on Data Engineering, ICDE 2005*. IEEE Computer Society.

[9] B. Godfrey, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica. Load balancing in dynamic structured P2P systems. In *Proc. IEEE INFOCOM*, Hong Kong, 2004.

[10] L. Gravano, H. Garcia-Molina, and A. Tomasic. GlOSS: text-source discovery over the internet. *ACM Trans. Database Syst.*, 24(2):229–264, 1999.

[11] V. King and J. Saia. Choosing a random peer. In *PODC*, pages 125–130, 2004.

[12] B. T. Loo, R. Huebsch, I. Stoica, and J. M. Hellerstein. The case for a hybrid P2P search infrastructure. In *Proceedings of the Third International Workshop on Peer-to-Peer Systems (IPTPS)*, La Jolla, CA, USA, 2004.

[13] L. Massoulie, E. L. Merrer, A.-M. Kermarrec, and A. Ganesh. Peer counting and sampling in overlay networks: random walk methods. In *PODC '06: Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, pages 123–132, New York, NY, USA, 2006. ACM Press.

[14] Medline database, US National Library of Medicine, 2006. http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?DB=pubmed.

[15] Medical subject headings, US National Library of Medicine, 2006. http://www.nlm.nih.gov/mesh/.

[16] L. Michael, W. Nejdl, O. Papapetrou, and W. Siberski. Improving distributed join efficiency with extended bloom filter operations. In *21st International Conference on Advanced Information Networking and Applications (AINA-07)*. IEEE Computer Society, 2007.

[17] 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 *Proceedings of the Eleventh International World Wide Web Conference (WWW2002)*, Hawaii, USA, May 2002.

[18] H. Nottelmann and N. Fuhr. Evaluating different methods of estimating retrieval quality for resource selection. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, New York, NY, USA, 2003. ACM Press.

[19] O. Papapetrou, W. Siberski, W.-T. Balke, and W. Nejdl. A combination of DHTs and peer clustering for distributed information retrieval. Technical report, L3S Research Center, 2007. Available online at `http://www.l3s.de/~papapetrou/publications/dhtclusters.pdf`.

[20] I. Podnar, M. Rajman, T. Luu, F. Klemm, and K. Aberer. Scalable peer-to-peer web retrieval with highly discriminative keys. In *23rd International Conference of Data Engineering (ICDE 07)*. IEEE Press, 2007.

[21] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content addressable network. In *Proceedings of ACM SIGCOMM 2001*, 2001.

[22] R. Siebes. pNear: combining content clustering and distributed hash tables. In *P2PKM*, 2005.

[23] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 Conference on applications, technologies, architectures, and protocols for computer communications*. ACM Press New York, NY, USA, 2001.

[24] S. Zoels, Z. Despotovic, and W. Kellerer. Cost-based analysis of hierarchical DHT design. In *Proceedings of the 6th International Conference on Peer-to-Peer Computing*, Cambridge, UK, 2006.