Aspects of Adaptivity in P2P Information Retrieval

Wolfgang Müller, Andreas Henrich and Martin Eisenhardt

Bamberg University, Bamberg , Germany,
wolfgang.mueller@wiai.uni-bamberg.de,
WWW home page: http://www.uni-bamberg.de/wiai/minf

Abstract. Peer-to-Peer networks are comprised of multiple independently administered computers (peers) that cooperate via a common protocol in order to achieve a goal common to the peers. Helping the user find relevant information in a P2P network is the subject of the field of Peer-to-Peer IR.
In order to be successful, a P2P-IR system needs to be adaptive in several respects. It has to adapt both to the user and to its environment. Within this article we detail the motivations and challenges of P2P-IR, as well as the ways in which P2P-IR systems adapt and where improvement is needed in order to achieve adaptive multimedia retrieval.

1 Introduction

Peer-to-Peer networks consist of multiple independently administered computers (peers) that cooperate with each other serving a goal that is common to the peers. The word *peer* indicates that the participants in the P2P network have equal rights and opportunities. In true P2P networks, there are no central components.

It is common grounds that there is the need to discover the network's resources in order to make use of a P2P network. This motivates the research into retrieval in P2P networks. For a couple of years, the focus lay on exact search in P2P networks, however, there is a growing interest in similarity search, *i.e.* information retrieval (IR) in P2P networks [25, 18, 16].

What is the motivation of such networks? In fact, currently, there is a growing proportion of self-generated media. Services like `myspace.com`, `blogger.com` or `flickr.com` all offer users the opportunity to put their opinion and their feelings into media objects, upload them to the site and then serve it to the world and have the result viewed and annotated for search by friends and strangers alike. In other words, there is a growing amount of data generated by end users for the use by end users.

On the other hand, Google and competitors offer search for the mainstream and increasingly also for specialized communities. However, there are limits to the current crawler-based system: a growing number of users is reluctant to give their personal data to huge data-collecting enterprises. At the same time crawlers reach their limit in the sense that many site owners of small sites complain that

too much of their traffic is due to visits of web site crawlers for search engines. This puts a limit to the freshness of data accessible via web search engines.

P2P-IR offers the promise of freshness of index data. Moreover, there is the hope that as each machine in the P2P network is responsible for comparatively few documents, there is the possibility to use sophisticated query processing methods that might be too costly for classical search engines such as Google, Yahoo or MSN search.

However, looking more closely, there are several challenges to P2P search. All of them are, in fact, linked to the need for adaptivity. We identify four main aspects of adaptation that a P2P-network has to perform. One of them is IR specific, the other three are P2P specific:

IR specific: P2P-IR inherits the adaptation problems from IR

User query behavior: The system has to adapt to the user and his information need, or more precisely to his perception of usefulness and relevance of media objects in the given query situation.

P2P specific: These adaptation problems are common to P2P systems.

User online behavior: Experience shows [11] that users of P2P networks have strongly differing behavior with respect to how long they stay online and how much data they share.

The word *churn* describes the fact that the population of a P2P network is constantly changing. We are speaking of *the* P2P network and *its* participants, but these words do not describe a P2P network well. There is as much *the population* of a P2P network as there is *the population* of a huge railway station (think: Paris, *Gare de l'Est*): The overwhelming majority of a railway station's population will be part of this population less then a quarter of an hour. However, some very few people will work at the station all day. Similarly, many measurement studies (*e.g.* [11]) in P2P networks report that many (up to $\approx 80\%$!) peers joining a P2P network stay less than one minute in the network. Evidently a P2P network has to adapt to this churn of population.

An important insight is that the churn present in P2P networks calls for restricted goals of availability. In the context of multimedia retrieval, it might be feasible to replicate indexing data, but it will be infeasible to replicate the actual documents [2].

Peer system properties: There are peers with widely differing computing power (*e.g.* from a 200MHz portable device to a 4GHz Pentium D) and network bandwidth (*e.g.* from a mediocre 40kbit/s telephone connection to 16Mbit/s DSL lines). P2P networks need to find the right compromise between fairly balancing the load and making network participation impossible for users of legacy equipment.

Attacks: Finally, participating peers can be contributors or attackers. In contrast to client/server networks where there is one data provider and many data consumers. If a client tries to alter a server's data, it is easy to tell who is the attacker and who is attacked. In P2P networks service consumers (*i.e.* peers) are on the same side of the fence as service

providers (*also* peers), so there is no easy way to tell if a contribution is legitimate or not.

In fact, all of the adaptation challenges have to do with diversity. Adapting to users means adapting to their differences. The same, P2P systems seek to *adapt to heterogeneity* of the P2P system and its environment. As we will see in the following, in some situations one can even *make use of the heterogeneity.*

1.1 Structure of the paper

In the following we will consider image Query by visual Example (QbvE) as an example for multimedia-retrieval, a variety of Content-Based Image Retrieval. While we are aware that QbvE is not the only way of querying multimedia data, we do assume that this way of query processing is representative and the basis for many more complex and more powerful query paradigms.

In classical QbvE, images are indexed by extracting a feature vector from each image and indexing the resulting collection of vectors for search. A query is processed by transforming the query into a feature vector $\boldsymbol{q}$ and by ranking the image in the collection by the distance $\delta(\boldsymbol{q}, \boldsymbol{v})$ of each feature vector $\boldsymbol{v}$ to $\boldsymbol{q}$. The best-ranked document is the document whose feature vector has the smallest distance to the query. Typically, only k documents are of interest.

In other words, the image query is mapped onto a ranked k-Nearest-Neighbor (k-NN) query between feature vectors.

However, there is an additional complication. Typically QbvE systems try to improve query performance by solliciting feedback from the user. Typically, the user can mark documents as relevant or irrelevant to the query. The system reacts by either modifying $\boldsymbol{q}$, or by *modifying the distance measure* δ. Especially the latter poses challenges to the indexing structure.

There is a large number of diverse approaches to performing such k-NN queries in P2P networks. Roughly, they can be sorted into three groups, namely

1. Replication in unstructured networks,
2. Approaches based on distributed hash tables, and
3. Routing by data summaries and source selection.

In the following we will describe these approaches, and we will describe how they realize adaptivity. Please note that while we find this classification useful, many systems use combinations of these approaches. Freenet, for example, performs replication as well as summary-based routing.

2 Replication in unstructured networks

Despite the existence of sophisticated techniques for using unstructured P2P networks (such as [22, 27]), the term *unstructured networks* typically is associated with the first generation of the Gnutella P2P protocol [6]. In a classical Gnutella network, peers are connected via TCP/IP connections. Each peer is connected

with a small number of *neighbors*. Each peer steadily discovers new neighbors in case its current neighbors leave the system. When receiving a query either from a user or from other peers, the receiving peer forwards the query to all its neighbors, except for the source of the query. On receiving the results it will forward these results to the source of the query, *i.e.* either to another peer or to the querying user. The querying user then can choose documents to download.

This method is simple and robust. However, it quite quickly hit the first scalability barrier. Every node receives every query. Nodes with a slow network connection eventually end up doing nothing but forwarding queries, and they are not able to serve or to request documents any more.

2.1 Adaptation to system diversity and to user online behavior

A first attempt at reducing the communication load is to limit the reach of queries via a so-called *Time To Live (TTL)*, effectively forwarding the query to just a (random) subset of the peers. However, this method still treats all peers equal and does not cater for the heterogeneity of the system.

The current method of choice is to introduce so-called *super nodes* or *super peers* [27]. These peers are more powerful and reliable than the average peer and take more responsabilities in the network: Each super peer is responsible for a set of normal peers. When a normal peer connects to a super peer, it will send a replicate of all its indexing data to the super peer. Subsequent queries will be handled by the super peer[1]. The normal peer just comes into play if it can contribute to the query result. This way, normal peers are shielded from the majority of the query traffic. To summarise: super peers act as servers for normal peers, and as classical Gnutella peers among each other.

Super peer architectures make use of the heterogeneity in P2P networks. There are peers that have more bandwidth than others, and there are peers that stay longer in the network than others. In fact, the peer online time distribution is such that it is safe to assume that a peer that has stayed an hour within the network will stay much longer in the network. So, the network elects peers as super peers that have stayed online a long time and that are willing to serve as super peers. Here heterogeneity helps making the choice.

2.2 Adaptation to the index data

In replication based networks, the algorithm makes sure that a query reaches all super peers. As all super peers combined contain all indexing data, each super peer just has to act like a non-P2P server: it processes the query locally and forwards a ranking to the querier.

Obviously, ranked similarity queries that can be processed in one centralized server can also be processed in P2P network with a super-peer architecture.

[1] Some systems only ship peer data summaries instead of the full indexing data.

2.3 Adaptation to the querying user

From the above follows that also complex relevance feedback queries can be processed using super-peer methods. In fact, this seems like an opportunity for building adaptive systems that support complex, interactive query processes.

However, when one looks at the actual query times needed to process a query in a Gnutella network, they are in the region of tens of seconds up to several minutes. The high latency between issuing and completing a query step is the main weakness of this type of architecture and currently makes it unsuitable for interactive query processes that use relevance feedback.

3 Distributed indexing structures

Distributed indexing structures try to get away from query processing that involves looking at all data points. As indicated by its name, the approach is similar to the approach of non-distributed indexing structures: The network maintains a structural invariant in the presence of peers joining and leaving. The data to be indexed is inserted at the proper position in the indexing structure. On processing a query, an algorithm finds the nodes that contain the index data needed.

The main advantage of distributed indexing structures is that they are conceptually very close to non-distributed indexing structures. Their main disadvantage in the P2P setting is that peers entering the network have to upload their index data up-front when entering the network.

Most current distributed indexing structures are based on *Distributed Hash Tables (DHTs)*. DHTs are one of the main architectural advances of current P2P research with respect to the initial Gnutella architecture.

In contrast to super-peer architectures that do not provide any guarantee of search quality, DHTs consisting of N peers are able to determine in $O(\log N)$ *hops*[2] if a data item is present in the network or not. The price for this precise knowledge is high: The large majority of DHTs does not support similarity search: The operations supported are the insertion of key/value pairs and the retrieval of a value given a key. The most prominent DHTs with these properties are Chord [24], Pastry [20] and Kademlia [14]. The latter has been successfully fielded in a large-scale consumer application: eDonkey.

Most DHTs identify each node using a long bit string without semantic meaning. In addition to identifying nodes such that they can be recognized even after a change of IP address, the identifier determines the position of the node relative to other nodes in the DHT. Media object keys in DHTs are also bit sequences of the same lengths as peer IDs. The P2P algorithm now assigns to each peer within the network a region in the space of possible bit sequences for which the peer will be responsible. When inserting a key/value pair into a DHT, a routing

[2] A *hop* is a step of indirection. If A sends a message to C via B, the message is routed over two *hops*.

algorithm will find the node responsible for the key and assign the key/value pair to it.

Chord, for example has a ring topology. Each peer has two neighbors, one with a smaller (right), one with a bigger ID (left). Each peer is responsible for keys that are smaller than or equal to its ID and bigger than the ID of its right neighbor. Using this architecture one would be able to find a given key in linear time. In order to achieve a speedup, each node maintains $O(\log N)$ connections across the ring, the *fingers*. Judicious use of these fingers enables each peer to look up any key in $O(\log N)$ time.

CAN DHTs (Content-addressable networks, [19]) work differently. Here the space of possible IDs are multi-dimensional real-valued vectors. Typically each vector component is limited to the interval $[0; 1)$. In CAN each peer is responsible for a rectangular region in key space. While in a classic CAN each peer is only connected to peers that are responsible for the regions neighboring its own key space, there exist modifications of CANs that build small-world networks on top of the classical CAN structure, obtaining $O(\log N)$ lookup time [8]. In contrast to Chord, CAN is able to perform efficient similarity search on vectors. However, due to the curse of dimensionality [26, 1], this ability is limited to small dimensionalities.

Both Chord and CAN have been used as building blocks for the design of IR-applications. In the following we will shortly describe two applications: PRISM [21], a Chord-based system, and pLSI [25], a CAN-based system.

PRISM indexes each vector $\boldsymbol{x}$ by placing $\boldsymbol{x}$ on a small number of nodes in a Chord DHT. Using the resulting distributed indexing structure, it can process k-Nearest-Neighbor (k-NN queries for high-dimensional vectors).

The placement of each vector in PRISM is calculated using distances to a fixed set of *reference vectors*. When processing a query, the node issuing the query $\boldsymbol{q}$ calculates the set of nodes where $\boldsymbol{q}$ would be placed and searches for similar vectors there, sending the nodes $\boldsymbol{q}$ as the query. The main innovation of PRISM is the algorithm for finding the nodes on which to place the data vectors.

In order to index a vector $\boldsymbol{x}$, the distance of $\boldsymbol{x}$ to a number n_r of reference vectors $\boldsymbol{r_i}$ ($i \in \{1, \ldots, n_r\}$) is calculated, yielding $\boldsymbol{\delta} := (\delta_1, \delta_2, \ldots, \delta_{n_r}) := (\delta(\boldsymbol{x}, \boldsymbol{r_1}), \ldots, \delta(\boldsymbol{x}, \boldsymbol{r_{n_r}}))$. Typically, $\boldsymbol{\delta}$ has fewer dimensions than $\boldsymbol{x}$. Now, one straightforward way to proceed would be to index $\boldsymbol{\delta}$ via a distributed vector indexing structure. The authors of PRISM, however, go a different way. In PRISM, the $\boldsymbol{r_i}$ are *ranked* by their similarity. The result of this ranking is a list of indices $\boldsymbol{\iota} = (\iota_1, \ldots, \iota_{n_r})$ such that $\boldsymbol{r_{\iota_1}}$ is the reference vector closest to $\boldsymbol{x}$, $\boldsymbol{r_{\iota_2}}$ the second closest and so on.

Then, pairs of indices are formed. The pair formation is a fitting parameter, the original PRISM paper suggests $\{\iota_1, \iota_1\}$ (*i.e.* storing the a pair consisting of twice the index of the best match) $\{\iota_1, \iota_2\}$, (the reference point index of the best match and the second best match), $\{\iota_2, \iota_3\}, \{\iota_1, \iota_3\}, \{\iota_1, \iota_4\}, \{\iota_2, \iota_5\}, \{\iota_2, \iota_4\}$, $\{\iota_3, \iota_4\}, \{\iota_1, \iota_5\}, \{\iota_4, \iota_5\}, \{\iota_3, \iota_5\}$ for their dataset. From each of the pairs a Chord key is calculated, and this key is used for inserting the vector $\boldsymbol{x}$ into the Chord ring.

Query processing works by finding out which peers would receive the query vector $\boldsymbol{q}$ if it was a new data item and forwarding the query vector to these peers. This involves again the calculation of index pairs, which we will call *query pairs* in the following. In order to reduce query processing cost, the query processor can choose to contact only nodes pertaining to only a subset of the query pairs. Doing this also reduces recall, one has to find a useful tradeoff.

pLSI pLSI [25] follows another approach that is more classical. Here Latent Semantic Indexing, *i.e.* a singular value decomposition [7] is performed in order to reduce the dimensionality of the vectors to be indexed. At the same time, the SVD achieves an ordering of the dimension *by their importance*. The remaining (still) high dimensional vectors are cut into low-dimensional slices. Each slice and the ID of the document it pertains to is entered as a key/document id (*i.e.* scliec/document id) pair into into a CAN. On receiving a query from its user, a peer cuts up the query vector into slices and then queries the CAN, starting with the most important dimensions. Results for several slices of each vector will be combined. As in PRISM we can process k-NN queries using pLSI.

3.1 Adaptation system diversity and to user online behavior

In terms of adaptation to system diversity and user online behavior, DHT-based systems inherit their properties from DHTs.

DHTs have the advantage of being provably efficient, and they can be tested in a data-independent manner. This property has made them a subject of extensive research. One focus of this research has been making DHTs churn resistant [23], and to introduce load balancing where the load balancing that is inherent to the DHT algorithms does not suffice [10].

DHTs perform replication of key/value pairs in order to ensure high availability. Obviously, if there is much index data (*i.e.* the values of the key/value pairs) stored in the network, the continuous replication alone will generate *much* traffic.

Example: Consider an inverted file index, in which each document of $N_{d,p} = 1000$ documents per peer is represented by $m = 1000$ vector components. Assume $r = 20$ fold replication. Consider that 5% of the peers is leaving every five minutes. In (very conservative) estimations we would count each vector component to be stored as $4bytes$. In our hypothetic but realistic setting, $0.05 \cdot N_{d,p} \cdot m \cdot r \cdot 4 = 4Mbyte$ would have to be shipped *per peer* every five minutes just to maintain the network. In other words, a peer participating 8 hours a day would have to send 10GB/month over the network just for participating.

Load balancing can be performed by having peers that hold too many keys distribute some of their keys to their neighbors.

Adaptation to differing processing power and network latency can be performed *e.g.* by sending each query to several nodes in the DHT. Query answering in DHT is a multi-step process in which the querying peer queries its neighbors for suitable next nodes that are closer to the wanted key. By sending the query to several nodes at the same time, the querier can choose the answer of the fastest

answering node for continuing the query process. By this, DHTs tend to sollicit more strongly nodes that are more performant.

3.2 Adaptation to the index data

In contrast to replication-based systems, DHT-based IR systems make use of the structure and distribution of the data they are indexing in order to create an efficient data structure. Our preliminary experiments (see Fig. 1) suggest that *e.g.* in PRISM the dimensionality of the features indexed matters, and that this influences the number of reference points that are to be chosen depending on the data.

Similarly, the usefulness of an LSI depends on the dimensionality of the feature set and thus on the type of data to be indexed. The outcome of an LSI is data dependent. So, before indexing a collection of vectors, a pLSI network needs to perform an LSI of the data to be indexed.

The same applies when data drift over time. From time to time, a pSearch network's administrator (or an algorithm that is not reported, yet) will have to decide to adjust network parameters in order to suit the new data distribution over the peers. Another method would be to perform such a readjustment periodically, avoiding difficult, and probably faulty decisions.

3.3 Adaptation to the user

As of yet, there is *no* research we are aware of that considers adaptation of DHT-based P2P-IR networks to user feedback.

Distributed inverted files are sufficiently similar to non-distributed inverted files to be able to support the processing of relevance feedback. However, more research is needed for evaluating if the cost of distributed processing of relevance feedback queries is within reasonable bounds. [13] suggest that naive use of inverted files in large networks is beyond reasonable communication cost even when proceeding queries with few query terms.

Distributed indexing structures for non-sparse real-valued vectors, such as PRISM and pLSI suffer from the fact that they assume one distance measure when filling the indexing structure. We would expect that the performance degrades when changing that distance measure, *e.g.* in order to respond to user feedback. Indeed experiments confirm slightly degrading performance. Please see the experimental section 5 for details.

4 Routing-based approaches

FIXME: hier ist die Nomenklatur nicht klar.

As distributed indexing structures, routing-based approaches seek to get away from considering all data vectors for each query. However, in contrast to distributed indexing structures, summary based approaches leave the bulk of the indexing data in the peers that hold the corresponding documents. Routing

based systems seek to improve the query performance by improving the network's topology and by creating routing tables that enable *semantic routing* between peers.

Creating routing tables for semantic routing involves creation of summaries of a peer's collection and shipping the collection summary to the right place in the network.

4.1 Freenet

Freenet, described in [5] performs routing by document keys. It has been extended for the use in (text) information retrieval by Kronfol [12]. To our knowledge, there is no extension of Freenet for the use of multimedia data. However, the techniques applied in Freenet have influenced other systems.

As Gnutella, Freenet is unstructured. However, Freenet queries are not forwarded from the querying node to all its neighbors. Instead, each node contains a routing table. The routing table contains a list of peer/document identifier pairs. A peer/document identifier pair p/id is entered in the routing table, if p has provided id in the past. In each node, a query will be routed to the p whose id matches most closely the query. If there is no possibility to route the query to a suitable next node, backtracking is performed. When the searched document is found, the document found will be sent back along the path of the query. Peers on this backward path of the query can choose to cache the document, and they can choose to enter themselves as the source of the result document.

4.2 DISCOVIR

A well-published system that uses summaries and topology improvements in order to perform Content Based Image Retrieval (CBIR) is DISCOVIR [18]. DISCOVIR indexes high-dimensional feature vectors. Each peer is summarised via an average feature vector $\boldsymbol{v_{p,avg}}$ and σ, the corresponding standard deviation. This summary is used in two ways:

Privileged *vs.* normal links: Each peer has two classes of links. (i) Normal links that work much like Gnutella links. (ii) *Privileged links* that build a second Gnutella-like networks between peers that are similar to each other. If they are similar to each other can be determined via their summaries.

Query filter: A peer that receives a query calculates the distance between the query vector $\boldsymbol{q}$ and the average vector of the peer and tests if the distance is below a threshold that is calculated relative to the standard deviation $||\boldsymbol{q} - \boldsymbol{v_{p,avg}}|| < c \cdot \sigma$. If the query is too far away from the average vector, the peer will not run the query on its local data. It will forward the query as described above.

Sia *et al.* report improvements with respect to classical Gnutella. However, to our knowledge, this method has not yet been tested with a realistic large-scale data distribution over peers.

4.3 PlanetP

DISCOVIR is insatisfactory in the sense a large fraction of peers has to be contacted in order to process a query.

In fact, the curse of dimensionality that makes the creation of successful distributed indexing structures difficult is also the primary source of difficulty when trying to implement multi-hop routing strategies for performing CBIR in P2P networks. Due to the curse of dimensionality, the summaries cannot be very selective, and thus cannot claim that a routing decision based on such a summary is correct with a high probability. However, multi-hop routing requires the routing decision to be correct with a very high probability.

Example: Imagine a scheme that routes a query to the peer holding the most similar vector $\boldsymbol{v_t}$ with respect to a query $\boldsymbol{q}$. The query would be routed over 20 hops. If we assume that the routing decision is correct 99% of the time, still this method will route $0.99^{20} = 80\%$ of all queries to the correct peer. However, if we assume routing decisions to be correct at 80% of the hops, the query will reach the node holding $\boldsymbol{v_t}$ only $\approx 1\%$ of the time!

PlanetP reacts to these considerations by employing *one-hop routing*, known in the area of distributed IR and databases as data source selection, inspired by classical methods of distributed information retrieval [9, 4]. In PlanetP each peer knows summaries of *all* other peers. Summaries are replicated via so-called *rumor spreading*. Obviously this scheme is not scalable as the number of peers to keep track of grows $O(N)$. We have presented a scalable version of PlanetP, Rumorama [16] whose properties will be discussed below but whose details are out of scope here.

A PlanetP-peer receiving a query $\boldsymbol{q}$ from its user (we will call this peer the *query peer*) will rank the other *peers* with respect to the query. The peers will be ranked by the probability that they contribute one or more documents to the result set of the query. After this peer ranking has been obtained, the query peer will contact the most promising peers, sending them $\boldsymbol{q}$. The peers contacted will process q using their local data store and then return the results to the query peer. The query peer will generate a combined ranking of peers.

PlanetP is specialized on text information retrieval. PlanetP uses Bloom filters [3] as summaries. Each summary describes which index terms are present in a given peer. Unfortunately Bloom filters are not adapted to the indexing of densely populated high-dimensional vectors.

We have presented work about peer data summaries based on *cluster histograms* for use in PlanetP-like networks. These summaries can be used for collections of images and have been tested on 166-dimensional histograms extracted from stock photos and consumer photos [17, 15].

In this method, first a *global k-means clustering* is derived over all the images present in the network. As shown in [15], such a clustering can be calculated efficiently, *i.e.* without having peers transfer their data collection. The result of the clustering is a set of *cluster centers* $\boldsymbol{c_i}$. Every peer j will now assign each of its vectors $\boldsymbol{v_k^j}$ to the closest cluster center, and it will count how many vectors

are assigned to which center. Doing this, it obtains a cluster histogram h_i^j that assigns to each cluster center $\boldsymbol{c_i}$ a document frequency given the peer j.

We are currently preparing a paper that presents and evaluates diverse peer ranking methods based on cluster histograms. The simplest method is the one described in [17] and will be presented here: When processing queries, the query peer first finds which center $\boldsymbol{c_i}$ is closest to $\boldsymbol{q}$. We call the closest center $\boldsymbol{c_{i_q}}$. The query peer then will rank the peers j by decreasing histogram value $h_{i_q}^j$ for the cluster center $\boldsymbol{c_{i_q}}$.

Adaptation to system diversity and to user online behavior The architectures presented here are very different in nature, and thus react very differently to user online behavior and system diversity.

The outcome of Freenet's caching scheme is that popular documents are cached at many peers in the network, and that the addresses of cached copies will be contained in many routing tables. So, after inception of the network, the network structure and query performance will adapt to the users need. Moreover, peers can choose if they want to keep long or short routing tables, if they want to enter themselves as data source, and if they want to cache documents. This enables peers to choose their load. In addition, popular documents will be cached all over the network and thus the load will be well-balanced.

While Freenet does not perform search on multidimensional vectors (neither does FASD), the interest of Freenet stems from the fact that similar techniques of gradual routing improvement are used for IR, but not for CBIR, yet.

DISCOVIR, in its pure form inherits most of the disadvantages of Gnutella. A peer that has privileged links to peers with often-queried images will tend to be queried often. DISCOVIR does not give possibilities to reduce the load, except to avoid building privileged links to other peers. In this case, the peer will be only rarely contacted.

DISCOVIR also inherits Gnutella's advantages. The peer data summaries are very small, they are shipped when discovering other peers, and thus a peer entering the network *gradually* improves its connectivity which is in contrast to DHT-based approaches in which there is no gradation between indexed and non-indexed data.

PlanetP is strongly churn-resistant (as every peer knows every other peer, it is very difficult to make the system break). However, it does not cater for system diversity. Every peer in the network ist supposed to stay informed about all the other peers in the network, *i.e.* all peers stay informed about the same number of other peers, incurring the same maintenance load. The actual query load of a peer depends on the summary that it is posting. If the summary makes the peer a probable holder of many highly demanded documents, it will be contacted more often than if it posts that it contains one single rarely demanded document.

Rumorama, a hierarchical variant of PlanetP enables balancing of the maintenance load. Rumorama introduces a Pastry-like hierarchy on top of the PlanetP network. Peers are enabled to choose with how many *friend nodes* they want to

exchange summaries. Still, a structured multicast algorithm enables considering summaries of all peers when processing the query.

4.4 Adaptation to the index data

As the indexing data in DHT-based approaches, summaries need to be tailored to the data types that are indexed. For some types of summaries there is also the need to recompute and redistribute the summaries depending on changes in the data collection.

4.5 Adaptation to the user

Both DISCOVIR and PlanetP are expected to suffer performance losses when the distance measure changes, as the summaries are adapted to a given distance measure. This is described in the following section.

5 Experiments

In Fig. 1 we show two experiments performed using the source selection based method described in [15] and using PRISM. By changing the distance measure at query time, we also *simulated* a relevance feedback query.

For our experiments on PRISM we re-implemented PRISM in a simulator. As we were interested rather in number of distance calculations than in the number of peers contacted[3]. We used a non-tuned simple PRISM version with randomly chosen reference vectors.

Experiments on PRISM were performed with a 1 million image `flickr.com` crawl. For our experiments on the source selection based method we also used a simulator, but with a 50'000 image `flickr.com` crawl. Data distribution over peers matters when using this method. We took the approach to model each peer's data by the data corresponding to one flickr user. This way, the 50'000 images were distributed unevenly over ≈2'600 peers. From all images in both collections we extracted 166-D color histograms using $18 \times 3 \times 3$ HSV color bins and 4 grey levels, as described in [15].

All curves in Fig. 1 plot the fraction of 20-NN found plotted against the amount of data points considered (*i.e.* distance calculations) per query. Curves reaching 1 more quickly correspond to better performance.

Note that in this experiment our non-tuned simple PRISM version actually performs *worse than scanning the whole collection once*. Please note that the results for PRISM measured here are much worse than the results presented in [21] using 80-D features, suggesting that dimensionality matters and that

[3] This makes sense as the number of peers contacted in PRISM depends largely on the amount of load balancing that is performed. If there is little load balancing, few peers are contacted and PRISM rather behaves like a super peer scheme using large super peers.

careful tuning for an application can greatly improve results. In these present experiments, the source selection method performs better than PRISM.

In order to get a first impression on the behavior of both methods when changing the distance measure used for query processing with respect to the distance measure used for indexing, we simulated a changing distance measure due to user feedback by not evaluating the distance over the complete vectors, but just the distance by projecting both query and document vector on the first 20 components. The query with the changed distance measure was evaluated on the *unchanged indexing data*. The source selection method [15] takes a severe performance hit under these conditions. Our cluster-based variant of PlanetP needed to evaluate up to three times as many distances δ with the modified distance measure with respect to the original distance measure. PRISM also takes a performance hit, but much less so. However, source selection still performs better than PRISM, and still performs better than random search.

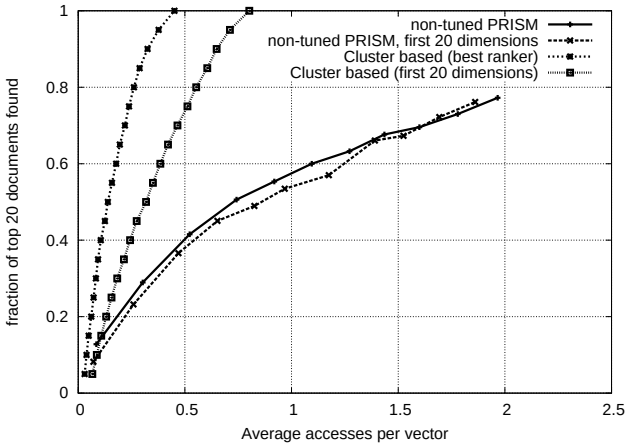

Fig. 1. Comparing PRISM and cluster-based summaries.

These experiments support our intuition that both summary-based methods and distributed indexing structures will have to undergo deeper tests if they are supporting adaptive multimedia retrieval. In order to be useful, the benchmarks applied need to be application driven and need to take the data distribution over peers into account.

6 Conclusion

We have presented examples for the main types of P2P architectures for the use in Multimedia Information Retrieval. We have chosen QbvE with relevance feedback as example application. Then we have described the adaptivity properties of some example systems.

Summarizing, one can state that P2P systems have reached an impressive state of the art in terms of load balancing and adaptation to churn. P2P systems can adapt well to challenging scenarios in which users stay only shortly in the network.

There is a useful baseline: Super-peer architectures easily enable any kind of k-NN queries. Their downside is that for processing a Super-peer query, all super peers need to be contacted. Other architectures, based on DHTs or on routing approaches seek to restrict the number of peers that need to be contacted for processing a query. However, their use for relevance feedback query processing is unclear. None of the methods described here has been tested for relevance feedback queries. We feel that this is an interesting open area of research.

References

1. C. C. Aggarwal, A. Hinneburg, and D. A. Keim. On the surprising behavior of distance metrics in high dimensional spaces. In *ICDT '01: Proceedings of the 8th International Conference on Database Theory*, pages 420–434, London, UK, 2001. Springer-Verlag.
2. C. Blake and R. Rodrigues. High availability, scalable storage, dynamic peer networks: Pick two. In *Ninth Workshop on Hot Topics in Operating Systems (HotOS-IX)*, pages 1–6, Lihue, Hawaii, May 2003.
3. B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7), 1970.
4. J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *Proc. 18th ACM SIGIR*, Seattle, Washington, 1995.
5. I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 46–66, July 2000.
6. Clip2. The Gnutella Protocol Specification v0.4. URL: `http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf`, 2000.
7. S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.
8. P. Ganesan, B. Yang, and H. Garcia-Molina. One torus to rule them all: multi-dimensional queries in P2P systems. In *WebDB '04: Proceedings of the 7th International Workshop on the Web and Databases*, pages 19–24, New York, NY, USA, 2004. ACM Press.
9. L. Gravano, H. Garcia-Molina, and A. Tomasic. GlOSS: text-source discovery over the internet. *ACM Trans. Database Syst.*, 24(2):229–264, 1999.
10. D. R. Karger and M. Ruhl. Simple efficient load balancing algorithms for peer-to-peer systems. In *SPAA '04: Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 36–43, New York, NY, USA, 2004. ACM Press.
11. A. Klemm, C. Lindemann, M. K. Vernon, and O. P. Waldhorst. Characterizing the query behavior in peer-to-peer file sharing systems. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 55–67, New York, NY, USA, 2004. ACM Press.

12. A. Z. Kronfol. FASD: A fault-tolerant, adaptive, scalable, distributed search engine, 2000.
13. J. Li, B. T. Loo, J. M. Hellerstein, M. F. Kaashoek, D. R. Karger, and R. Morris. On the Feasibility of Peer-to-Peer Web Indexing and Search. pages 207–215.
14. P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric, 2002.
15. W. Müller, M. Eisenhardt, and A. Henrich. Fast retrieval of high-dimensional feature vectors in P2P networks using compact peer data summaries. *Multimedia Syst.*, 10(6):464–474, 2005.
16. W. Müller, M. Eisenhardt, and A. Henrich. Scalable summary based retrieval in P2P networks. In O. Herzog, H.-J. Schek, N. Fuhr, A. Chowdhury, and W. Teiken, editors, *CIKM*, pages 586–593. ACM, 2005.
17. W. Müller and A. Henrich. Fast retrieval of high-dimensional feature vectors in P2P networks using compact peer data summaries. In N. Sebe, M. S. Lew, and C. Djeraba, editors, *Multimedia Information Retrieval*, pages 79–86. ACM, 2003.
18. C. H. Ng and K. C. Sia. Bridging the P2P and www divide with discovir - distributed content-based visual information retrieva. In *Poster Proc. of The 11th Interational World Wide Web Conf.* to be published, 2003.
19. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *Proc. 2001 Conf. on applications, technologies, architectures, and protocols for computer communications*, San Diego, CA, United States, 2001.
20. A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proc. 18th IFIP/ACM Intl. Conf. on Distributed Systems Platforms (Middleware)*, Heidelberg, Germany, 2001.
21. O. D. Sahin, A. Gulbeden, F. Emekci, D. Agrawal, and A. E. Abbadi. PRISM: indexing multi-dimensional data in P2P networks using reference vectors. In *MULTIMEDIA '05: Proceedings of the 13th annual ACM international conference on Multimedia*, pages 946–955, New York, NY, USA, 2005. ACM Press.
22. N. Sarshar, P. O. Boykin, and V. P. Roychowdhury. Percolation search in power law networks: making unstructured peer-to-peer networks scalable. In *Proceedings of Fourth International Conference on Peer-to-Peer Computing*, pages 2–9. IEEE, August 2004.
23. T. R. Sean Rhea, Dennis Geels and J. Kubiatowicz. Handling churn in a dht. Technical Report UCB/CSD-03-1299, EECS Department, University of California, Berkeley, 2003.
24. I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proc. ACM SIGCOMM Conf.*, San Diego, CA, USA, 2001.
25. C. Tang, Z. Xu, and M. Mahalingam. pSearch: Information retrieval in structured overlays. In *First Workshop on Hot Topics in Networks (HotNets-I)*, Princeton, NJ, 2002.
26. R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proc. Intl. Conf. on VLDB*, New York, USA, 1998.
27. B. Yang and H. Garcia-Molina. Designing a super-peer network. In U. Dayal, K. Ramamritham, and T. M. Vijayaraman, editors, *ICDE*, pages 49–. IEEE Computer Society, 2003.