Efficient content-based P2P image retrieval using peer content descriptions

Wolfgang Müller, Martin Eisenhardt, Andreas Henrich
Department for Applied Computer Science, University of Bayreuth

ABSTRACT

Peer-to-peer (P2P) networks are overlay networks that connect independent computers (also called nodes or peers). In contrast to client/server solutions, all nodes offer *and* request services from other peers in a P2P network. P2P networks are very attractive in that they harness the computing power of many common desktop machines and necessitate little administrative overhead. While the resulting computing power is impressive, efficiently looking up data still is the major challenge in P2P networks. Current work comprises fast lookup of one-dimensional values (Distributed Hash Tables, DHT) and retrieval of texts using few keywords. However, the lookup of multimedia data in P2P networks is still attacked by very few groups. In this paper, we present experiments with efficient Content Based Image Retrieval in a P2P environment, thus a P2P-CBIR system. The challenge in such systems is to limit the number of messages sent, and to maximize the usefulness of each peer contacted in the query process. We achieve this by distributing peer data summaries over the network. Obviously, the data summaries have to be compact in order to limit the communication overhead. We propose an CBIR scheme based on a compact peer data summary.

This peer data summary relies on cluster frequencies. To obtain the compact representation of a peer's collection, a global clustering of the data is efficiently calculated in a distributed manner. After that, each peer publishes how many of its images fall into each cluster. These cluster frequencies are then used by the querying peer to contact only those peers that have the largest number of images present in one cluster given by the query. In our paper we further detail the various challenges that have to be met by the designers of such a P2P-CBIR, and we present experiments with varying degree of data replication (duplicates of images), as well as quality of clustering within the network.

1. INTRODUCTION

Peer-to-peer (P2P) networks are overlay networks that connect independent computers (also called nodes or peers) in order to provide a common service. Although Peer-to-Peer systems (P2P systems) have gained popularity as file sharing systems, suitable retrieval techniques for text and multimedia data in P2P systems are still an open area of research. P2P Content Based Image Retrieval poses broad challenges, as in addition to the obvious distributed indexing problem the problem of fail safety and security awaits being solved. Within this paper, we will focus on retrieval performance, in failure free environments, as well as in environments with node failures, leaving the problem of organized attacks aside.

Our goal is a system which enables a user to share image data with other users via a P2P network. The P2P network should enable its users to search for visually similar images within the network by visual example (*i.e.* an example image). The system should then seek the P2P network for similar images and present a number k of result images to the user, just like a centralized CBIR system. The principles of such a system would be close to that of a centralized CBIR system: on indexing, images are transformed into feature vectors. The query by example is a k-next-neighbour (k-NN) query in the feature vector space.

The above scenario is difficult to put into reality, as there are the following constraints:

Self-adminstration: We want peers to be independent computers administered by independent persons. In particular, we want to place the bar very low concerning the capacities of the administrators of a peer.

Low reliability of each peer: We demand from participating peers low network reliability. That is, if a peer fails without notice, the rest of the P2P network should remain functional.

Internet Imaging V, edited by Simone Santini, Raimondo Schettini,
Proc. of SPIE-IS&T Electronic Imaging, SPIE Vol. 5304

Low bandwidth: While participating peers are *allowed* to have a high bandwidth, peers with a low-bandwidth internet connection should also be able to participate in the network.

High reactivity: We want the indexing data to reflect closely the current state of the network. Nodes joining or leaving the network should be able to use the system quickly, and the data they provide should be known to the system quickly.

In addition to these constraints, Indexing of vectors becomes very hard if the dimensionality of the feature space becomes very high. This makes it hard to create indexing structures that do better than linear time in the number of vectors indexed on collections of high-dimensional data. This fact is known as the so-called *Curse of Dimensionality.*

There are three main design parameters of a peer-to-peer network: *Firstly,* the *topology* of the network, *secondly* the *placement of data* within the network, *thirdly,* the *routing protocol* according to which messages traverse a network with given topology. Each of these parameters imposes a tradeoff between ease of administration, robustness, efficiency, and quality of service (*i.e.* quality of retrieval results) of the network.

We describe tests with a system that focuses mainly on providing efficient routing when processing k-NN queries for high-dimensional feature vectors. Our system is based on distributed clustering of the complete collection of the P2P network. Cluster histograms (one histogram for each peer) describing the local collection of each peer are distributed throughout the network. After this distribution process each peer knows the cluster histograms of all peers within the network. On query execution, the peer whose user issues the query will first locally seek the cluster histograms stored. It will then issue queries only to the peers likely to contain interesting data. The efficiency gain comes from the fact that the clustering only has to be performed very rarely, but that for each query only a fraction of the total number of peers has to be queried.

In this work, we extend our previous work described [8]. The new contribution is that we investigate in more detail the properties of the cluster histogram representation, namely with respect to the quality of the clustering, the number of clusters employed, and the degree of replication of the collection. In addition to that the current measurements where done in a distributed fashion, which enabled us to use a 2.7 times larger collection, as well as 3.6 times more peers for our experiments.

The paper is organized as follows: in the following section, section 2, we will describe CBIR and P2P systems, as well as our approach to P2P-CBIR. In section 3 we describe the distributed clustering algorithm used in our system. Section 4 covers the experimental setup. Section 5 shows the results, followed by the conclusion in section 6.

2. FROM P2P AND CBIR TO P2P-CBIR

The best known P2P systems, like for example Gnutella [2], offer search only for filenames in simple, but robust P2P-architectures. However, there quickly emerged a community looking for P2P network topologies and algorithms that provide guaranteed quality of service (*i.e.* quality of results) and efficiency. The perhaps most prominent examples are distributed hashtables (DHTs) that provide storage and lookup of key-value pairs with logarithmic $\mathcal{O}(\log N)$ or $\mathcal{O}(\sqrt[m]{N})$ complexity, given a network with N nodes of degree m [10, 12].

What does complexity mean here? Complexity is determined by four factors:

1. The number of messages to be sent and received by each peer.
2. The number of times a message is forwarded within the network (called the number of *hops* of a message). Each forwarding causes delays by network latencies.
3. The number of bytes sent by each peer.
4. The processor load placed on each peer.

In most cases currently treated in the literature, the processor load is negligible.

It must be noted that it does not suffice to analyze the complexity of query processing alone. We have to take into account the *life cycle* of each peer. We have to consider the following phases of the life cycle:

1. *Initial introduction:* The peer enters a network for the first time.
2. *Join:* The peer enters a network after having been offline.
3. *Operation:* The peer issues and answers queries.
4. *Leave:* The peer leaves the network.

There is a community trying to bring database and retrieval functionality to P2P networks by proposing methods that make useful tradeoffs between robustness, quality of service, and efficiency in all phases of the peer's life cycle. In the P2P-IR (Information Retrieval) community, we can distinguish two main approaches. The *first* tries to minimize the number of peers queried in a query process [3,6,9]. Along with this, the number of hops is also minimized. The *second* approach uses DHTs as a basis for the construction of distributed index structures such as inverted files [13].

The latter approach is very attractive in that it produces search results that are provably equal to those that would be obtained in a centralized setting with the same data collection. However, there are concerns about the viability of the approach: In very large networks with very large collections, systems will use too much bandwidth for handling queries with current internet technology [7].

We suggest the former approach, *i.e.* improving the routing of messages. Here, the general approach is to create routing tables in each peer, in which the routing table contains information about the data present in some peers.

One extreme approach in this sense is the search employed in PlanetP [3]. Here, each peer keeps information about *all other peers.* We will call this information the *peer data summaries.* Queries are processed by a three-step approach:

1. Compare all locally stored peer data summaries to the query. Obtain a ranking of peers.
2. Directly query all peers likely to contain useful results starting with the highest-ranked peers. Note: The peers contacted in this step will *not* forward any queries to other peers.
3. Merge results from other peers into one common query result.

The PlanetP peer representations are geared very much towards use in Information Retrieval. They are unusable for CBIR. In the following, we will present a peer data representation suitable for CBIR that can be used in a PlanetP-like scenario.

2.1. Probabilistic peer selection for CBIR

Vasconcelos [14] views building an image retrieval system as building a mapping from images to image classes. Let $\mathbf{x} \in \mathcal{X}$ denote an image feature vector and let $y \in \mathcal{Y}$, with $\mathcal{Y} = \{1, \ldots, M\}$, be the label of an image class. $\mathbf{X}, Y$ are the corresponding random variables. An image retrieval system is thus a mapping g from images to image classes:

$$\begin{array}{rccc} g: & \mathcal{X} & \rightarrow & \{1, \ldots, M\} \\ & \mathbf{x} & \mapsto & y \end{array} \tag{1}$$

Now, for a query $\mathbf{x}$ the system tries to minimize the retrieval error, that is the probability of retrieving images from a class different from that to which the query $\mathbf{x}$ belongs.

The retrieval error is minimized using the Bayes classifier

$$g^*(\mathbf{x}) = \arg\max_i P(Y=i|\mathbf{X}=\mathbf{x}) \quad (2)$$

$$= \arg\max_i P(\mathbf{X}=\mathbf{x}|Y=i)P(Y=i) \quad (3)$$

Similarly, we also can view the peer selection problem as a classification problem:

$$g^*(\mathbf{x}) = \arg\max_a P(A=a|\mathbf{X}=\mathbf{x}) \quad (4)$$

$$= \arg\max_a P(\mathbf{X}=\mathbf{x}|A=a)P(A=a) \quad (5)$$

$$(6)$$

Here, $P(\mathbf{X}=\mathbf{x}|A=a)$ is the probability of finding $\mathbf{x}$ in a given peer a, and $P(A=a)$ is the *prior probability*, which is proportional to the number of documents contained in peer a in our case.

2.2. Clusters of interest *vs.* independent identically distributed samples

Please note that from Eq. 6 follows that $P(A=a|\mathbf{X}=\mathbf{x})$ has to differ from peer to peer for a given $\mathbf{x}$; otherwise selecting a peer a is not possible as all alternatives have the same error probabilities. So the following has to hold for most pairs of peers a, b:

$$a \neq b \Rightarrow P(A=a|\mathbf{X}=\mathbf{x}) \neq P(A=b|\mathbf{X}=\mathbf{x}) \quad (7)$$

In particular, this means that if the data collections of the peers are independent identically distributed (i.i.d.) samples of the collection of the whole network, peer selection cannot work. This fact has to be taken into consideration when designing experiments for the system evaluation.

In our scenario, we assume that each peer is run by a user that has a small number of *interests*. The documents shared using the peer reflect these interests. Thus we assume that the documents shared in each peer belong to $N_{PeerClusters}$ *clusters of interest* specific for each peer. Our evaluation presented in sections 4 and 5 reflects this view.

2.3. Cluster histograms for peer selection

In [8] we compared two representations of peer data for routing. The best method in our experiments was a method based on clusters as described in this section:

We propose to divide the feature space into $N_{GlobalClusters}$ clusters, using the clustering algorithm described in the next section, section 3. We call the set of clusters $\mathcal{C}$. In general we can express the probability to find a document $\mathbf{x}$ in peer a as the product of the probabilities to find a cluster c in peer a, and the probability to find a document $\mathbf{x}$ in cluster c as is shown in Eq. 8:

$$P(\mathbf{X}=\mathbf{x}|A=a) = \sum_{c\in\mathcal{C}} P(\mathbf{X}=\mathbf{x}|C=c, A=a)P(C=c|A=a) \quad (8)$$

$$= \sum_{c\in\mathcal{C}} P(\mathbf{X}=\mathbf{x}|C=c)P(C=c|A=a) \quad (9)$$

Eq. 9 holds, if the cluster membership of $\mathbf{x}$ in cluster c is not influenced by its peer membership in peer a.

In our case, each $\mathbf{x}$ belongs to exactly one cluster $c_\mathbf{x}$. This greatly simplifies Eq. 9:

$$P(\mathbf{X}=\mathbf{x}|A=a) = P(\mathbf{X}=\mathbf{x}|C=c_\mathbf{x})P(C=c_\mathbf{x}|A=a) \quad (10)$$

$$= 1 \cdot P(C=c_\mathbf{x}|A=a) \quad (11)$$

That is, we simply have to look at how many documents in a given peer fall into $c_\mathbf{x}$, and we obtain an approximation of $P(\mathbf{X}=\mathbf{x}|A=a)$.

2.4. Fitting the peer data summaries into the system

In the previous section 2.3 we have described a peer data summary that can be used in a PlanetP-like query scenario. That means for the life cycle of a peer within our network:

1. *Initial introduction, Join:*
   (a) *Maintain clustering:* Joining the network will change the clustering of the network. In our current experiments described below, we ignore this fact. However, we do show measurements that evaluate how much additional peers without adjusting the clustering alter the performance.
   (b) *Learning peer data summaries:* On initial introduction, a peer will seek to learn the peer data summaries from all other peers in the network. Not the fastest, but the simplest way is to simply wait until the descriptions of all other peers are spread as rumours to the new peer. In the same fashion, the new peer will spread its own new data as rumour to its neighbours.
2. *Operation:* When processing a similarity query with the query vector $\mathbf{x}$, the peer will go through the following steps:
   (a) Determine the cluster $c_{\mathbf{x}}$ wo which the vector $\mathbf{x}$ belongs.
   (b) Rank the peers by decreasing $P(\mathbf{X}=\mathbf{x}|A=a)P(A=a)$.
   (c) Query the peers, starting at the lowest rank.
   (d) Merge the results obtained.
3. *Leave:* On leaving the network, the peer does not need to give any notification. (Note that whenever a peer contacted in step 2.c. does not answer, we simply proceed further in the ranking.)

3. P2P CLUSTERING OF VECTORS

In the previous section we have described how peer data can be summarized compactly and usefully using cluster histograms. It still remains to be described how such a clustering can be obtained efficiently in a distributed manner. This description follows in this section.

In short, we do clustering by a combination of the k-means clustering algorithm and the probe/echo algorithm for distributing messages within a network. In the following, we describe both algorithms in detail. The resulting algorithm is an extension for P2P networks of the algorithm described in [4].

3.1. k-means clustering

In k-means clustering clusters are determined by their cluster centroids. Given a point in space, it is assigned to the cluster whose cluster centroid is closest to it. When performing the clustering, an initial guess of a set $C \subset \mathbb{R}^n$ of κ cluster centroids is refined until the refined cluster centroids fulfill a finishing criterion. A typical finishing criterion is that the refined cluster centroids differ only very little from the cluster centroids of the previous refinement iteration. The name of the k-means clustering method stems from the refinement method: the new cluster centroids are determined by calculating the arithmetic mean of the member vectors in each cluster. Details are shown in the box named Algorithm 1.

The fact, that the refinement of cluster centroids is based on calculating the arithmetic mean of all vectors belonging to one cluster makes k-means clustering inherently data parallel (see [4]), *i.e.* the problem can be easily divided into parts, each part needing only parts of the whole data for a complete solution.

In principle the proposed method works as follows: We propose that each peer keep its local data local. Given a set of centroids for the complete collection (centroids for all vectors from all peers) $C_{old,total}$, a peer p is able to calculate a new, improved clustering $C_{new,p}$. The new centroids for the complete collection $C_{new,total}$ can then be obtained in a very simple fashion from the $C_{new,p}$ of all peers p. This can be done by a simple weighted summation, as is shown in the following simple two-peer, one-centroid case:

Algorithm 1: The k-means clustering algorithm.

Input : $D \subset \mathbb{R}^n$ – the set of data points $d_i \in D$.
Input : κ – the number of clusters the algorithm should calculate.
Output : $C \subset \mathbb{R}^n$ – the set of cluster centroids $c_j \in C$.
Data : $M \subset 2^D$ – a set of κ sets m_j where each m_j contains the documents d_i closest to c_j.

begin
  $MSE \leftarrow \infty$, $MSE_{old} \leftarrow \perp$;
  repeat
    foreach $m_j \in M$ **do** $m_j \leftarrow \emptyset$; // initialize all clusters as empty sets
    foreach $d_i \in D$ **do**
      Find closest $c_j \in C$;
      $m_j \leftarrow m_j \cup \{d_i\}$; // assign each object to its nearest cluster
    foreach $c_j \in C$ **do** $c_j \leftarrow \frac{1}{|m_j|} \sum_{d_i \in m_j} d_i$; // recalculate cluster centroids
    $MSE_{old} \leftarrow MSE$;
    $MSE \leftarrow \sum_j \frac{1}{|m_j|} \sum_{d_i \in m_j} (dist(d_i, c_j))^2$; // calculate cluster quality
  until $|MSE_{old} - MSE| < \varepsilon$
end

If we assume n multidimensional data items d_i allocated to some cluster; in the centralized case, the cluster centroid c_{total} would then be

$$c_{total} = \frac{1}{n} \sum_{i=1}^{n} d_i$$

If the first l data items are located on a peer a, and the remaining $n - l$ data items are located on a peer b, then we get two cluster centroids

$$\begin{aligned} c_a &= \frac{1}{l} \sum_{i=1}^{l} d_i \\ c_b &= \frac{1}{n-l} \sum_{i=l+1}^{n} d_i \end{aligned}$$

Then, the weighted mean c_{mean} of these two locally computed centroids c_a and c_b is

$$\begin{aligned} c_{mean} &= \frac{1}{n} \cdot (l \cdot c_a + (n-l) \cdot c_b) \\ &= \frac{1}{n} \cdot \left(l \cdot \frac{1}{l} \sum_{i=1}^{l} d_i + (n-l) \cdot \frac{1}{n-l} \sum_{i=l+1}^{n} d_i \right) \\ &= \frac{1}{n} \cdot \left(\sum_{i=1}^{l} d_i + \sum_{i=l+1}^{n} d_i \right) \\ &= \frac{1}{n} \sum_{i=1}^{n} d_i = c_{total} \end{aligned}$$

The above formula is easily extended to a multi-peer case, suggesting a three-step approach

1. An arbitrary number of peers p calculate their local centroid c_p
2. The c_p are to be merged into c_{total} by simply calculating a weighted mean, the weights being the *local collection sizes* of the peers.
3. Repeat, until the quality criterion of clustering is met.

Note that from the above consideration it can easily be deduced that the merging of peer cluster centers into cluster centers of the total collection can be done hierarchically: we do not need *one* central distance merging *all* results from *all* peers. We can do the merging along the edges of a spanning tree spanning the P2P network. At each level of the tree results are merged into a preliminary result that is merged with further preliminariy results. The hierarchical merging described here fits very well with the properties of the Probe/Echo algorithm described in the next section.

3.2. Probe/Echo

The Probe/Echo algorithm [1] is the concurrent equivalent of depth first search (DFS) in general graphs. A P2P overlay network is such a graph.

In a Probe/Echo process, the initiator marks itself as `engaged`, and as `initiator`. At the same time it sends a `PROBE` to all its neighbours. On receiving a `PROBE` from one neighbour (the node's *predecessor*), a node that is not yet `engaged` marks itself as `engaged` and sends `PROBE`s to all its neighbours, except its predecessor. An `engaged` node will count, but otherwise ignore any other `PROBE` it receives. If a node has received either a `PROBE` or an `ECHO` message from each of its neighbours, it will send an `ECHO` message to its predecessor. The Probe/Echo process is finished, if the `initiator` has received `ECHO`s from each of its neighbours.

If we see our graph as a set of nodes or vertices V and edges E: $G = (V, E)$, then the message complexity of the `PROBE/ECHO` process is $2|E|$, as each edge is used for sending either two `PROBE`s, or one `PROBE` and an `ECHO`.

3.3. Clustering using Probe/Echo

Our clustering algorithm depicted in the box named Algorithm 2 is a combination of k-means clustering and the Probe/Echo mechanism. On clustering, the peer initiating the clustering, becomes an `initiator`. It sends its current guess of cluster centroids along with a `PROBE` to its neighbours. Then the initiaor performs a k-means refinement of the current cluster centroids using its local data. Each of the nodes receiving a `PROBE` will do the same as the initiator, *i.e.* pass on the `PROBE` and perform a local refinement run. The results from the local refinement runs are merged via the `ECHO` messages. The `ECHO` message from peer p comes with refined cluster centroids C_p and weights W_p, enabling the receiving peer to merge these refined cluster centroids with its local cluster centroids and weights C_l and W_l, to obtain new C_l, W_l. When a given peer has received either `PROBE`s or `ECHO`s from all its neighbours, it will pass its then current C_l and W_l to its predecessor.

Please note that the number of centroids that is carried by either `PROBE` or `ECHO` messages is always equal to the number of clusters κ we want to obtain, *i.e.* it does not depend on the size of the data collection nor the size of the P2P network.

4. EXPERIMENTAL SETUP

We ran our experiments on 360 peers running on 20 standard PCs in our university's student software labs. So, on each PC there was running an average of 18 peers.

In our experiments we used a central brokering server for setting up a set of peers with known properties, as well as for collecting the results. However, the calculations themselves were all carried out in a distributed manner.

4.1. Creating connectivity, simulating replication and clusters of interest

Obviously for doing measurements, we need reproducible conditions. In this section we show how we created reproducible conditions for the main parameters affecting performance.

Algorithm 2: Our distributed clustering algorithm.

Input : *Neighbours* – the set of all neighbouring nodes.
Input : $D \subset \mathbb{R}^n$ – the data points $d_i \in D$ on the local peer.
Data : (C_p, W_p) – remote set of cluster centroids and associated weights.
Data : (C_l, W_l) – local set of cluster centroids and associated weights.
begin
  receive (PROBE, C_p) **or** (ECHO, C_p, W_p) **from** $p \in$ *Neighbours*;
  if PROBE **then**
    if $\neg$*engaged* **then**
      engaged $\leftarrow$ true; *received* $\leftarrow 1$; *pred* $\leftarrow p$; // received first PROBE
      send (PROBE, C_p) **to** *Neighbours*$\backslash\{pred\}$;
      $(C_l, W_l) \leftarrow kmeans(C_p, D)$; // one local iteration of k-means
    else
      received $\leftarrow$ *received* $+ 1$; // received additional PROBE
  if ECHO **then**
    received $\leftarrow$ *received* $+ 1$; // received ECHO
    $(C_l, W_l) \leftarrow mergeResults(C_l, W_l, C_p, W_p)$; // merge local and received results
  if *received* $= |Neighbours|$ **then**
    engaged $\leftarrow$ false; // termination condition fulfilled
    if *initiator* **then**
      terminate;
    else
      send (ECHO, C_l, W_l) **to** *pred*;
end

Connectivity of the peer network: As shown in [5], the speed and the cost of clustering depends on the number of neighbours each peer communicates with.

In our experiments each peer had between 3 and 6 neighbours. This is quite simple to assure: we build a P2P network starting with a set of unconnected nodes. Each peer that has not enough neighbours will request a list of potential new neighbours. The broker will give out addresses of peers as potential new neighbours whose number of neighbours does not exceed the limit imposed.

Clusters of interest: Simulating clusters of interest is nontrivial. This is due to the fact that we need two ways of clustering. *Firstly,* we need some clustering of the data to determine the clusters of interest that are present within each peer (see section 2.2). *Secondly,* we need some way of simulating clusters of interest within the peers.

Of course, it would be tempting to perform just one clustering and use it twice. However, in our view this would dramatically reduce the value of the experiments as the results would be unrealistically good.

For this reason we used an algorithm for assigning documents to peers as described in the box entitled Algorithm 3. It is a distributed modification of the algorithm presented in [8]. After an application of this algorithm on a P2P network, the number of documents per peer varies, which changes slightly some considerations with respect to [8].

Algorithm 3: Assigning images to peers

Data : $\mathcal{I}$ // all images in the collection
Data : $\mathcal{I}^*$ // all images not yet assigned to a peer
Data : $\mathcal{A}$ // a set containing all peers
Result : $\mathcal{M} : \mathcal{A} \rightarrow 2^{\mathcal{I}}$ // mapping peers to image sets

$\mathcal{I}^* \leftarrow \mathcal{I}$;
for $a \in \mathcal{A}$ /* each peer */ **do**
  $\mathcal{M}[a] \leftarrow \varnothing$;
  for $1 \leq b \leq N_{PeerClusters}$ /* each peer cluster */ **do**
    $\mathbf{i} \leftarrow$ **random element from** $\mathcal{I}^*$ // the peer retrieves a random image from the broker;
    $\mathcal{M}[a] \leftarrow \mathcal{M}[a] \cup \{\mathbf{i}\}$;
    $\mathcal{I}^* \leftarrow \mathcal{I}^* \setminus \{\mathbf{i}\}$;
    for $\mathbf{x} \in \left\{ \frac{n_{Docs/Peer}}{N_{PeerClusters}} - 1 \right.$ **best matches to** $\mathbf{i} \left.\right\}$ /* ranking obtained in peer*/ **do**
      if x *not yet assigned to any peer /* information from the central broker */* **then**
        $\mathcal{M}[a] \leftarrow \mathcal{M}[a] \cup \{\mathbf{x}\}$;
        $\mathcal{I}^* \leftarrow \mathcal{I}^* \setminus \{\mathbf{x}\}$;

Replication of data: Increasing the replication of data, *i.e.* increasing the number of times a particular piece of data is present in the network is supposed to improve retrieval performance, because the probability of finding useful documents by chance increases.

Within the above framework, replication can easily be simulated. For example, if we assume every item of the database to be present in the network *degree* times, we simply use algorithm 3, however not starting with an image collection $\mathcal{I}$, but rather with a multiset $\mathcal{I}_{degree}$ containing each image of $\mathcal{I}$ *degree* times.

4.2. Data collection

All experiments were performed using 166 bin color histograms (corresponds to 18 Hue × 3 Saturation × 3 Value bins for color and 4 bins for grey levels) calculated from COREL stock photos as described in [11].

4.3. Parameters measured

A list of four factors determining the efficiency of P2P solutions is given already in section 2. In our experiments we focused on the following two factors

1. The number of messages to be sent by each peer.
2. The number of bytes sent by each peer.

That is, we do not consider the number of hops (in a PlanetP routing scenario, there are no hops) and the processor load, and we focus on the essential parameters of communication load.

We performed efficiency measurements of two stages

1. Distributed calculation of a global clustering using the algorithm described in section 3.
2. Spreading the full peer descriptions over the network using one `PROBE/ECHO` run.

In addition to that, we measured the performance of the peer data summaries: We measured, how many peers we can discard on average using the peer data summaries. This was done as follows:

1. We randomly chose a query image **i** from the image collection.

2. We did a centralized query for **i** on the collection, obtaining the 20 best matches M_{20}.

3. We queried the peers one peer after the other starting with the peers most likely to contain result images according to the peer content representations. This access order of the peers gave us a ranked list $R = (r_1, r_2, \ldots, r_{N_{Peers}})$ of all N_{Peers} peers.

4. We did some statistics on the number of the images out of M_{20} found on the peers with rank r_j, as described in the following.

The performance measure we use is the *median peer rank*: R was scanned, and we noted the peer ranks of the peers that contained each of the M_{20} images, obtaining a list L. For example, when the query image $\mathbf{x}$ itself was found in the 4th-ranked peer the pair $(\mathbf{x}, 4)$ was added to L. Then the median of the peer ranks in L was calculated. Obviously, in the case of a random ranking of peers, the median peer rank for peers containing an image out of M_{20} would be $\frac{N_{Peers}}{2}$ if the number of documents per peer is equal among all peers, less otherwise.

That is, the median peer rank is a measure of how effective our peer data summaries are.

5. EXPERIMENTAL RESULTS

In our experiments, we distributed a subset of 50000 images of the Corel collection in varying degrees of replication over the network. The number of documents varied between peers. Typically, more than 27000 images were assigned to the peers. The fact that the number of documents varies between peers changes our baseline. We want to compare the performance of our system to the case of random search. In the case of random search on equal-sized peer data collections, we would expect to find one given data item in the $\frac{N_{Peers}}{2}$-th peer. In the case of non-equal sized peers we can sort the peers by their collection sizes, and visit the ones with large collections first, as by their size they are more likely to contain any given data item. For a typical data distribution within our experiments, finding a given data item would be possible within the first $0.4 \cdot N_{Peers}$ in this baseline case.

For each series of measurements, we first distributed the data over the peers (as described in the previous section). We then performed a distributed clustering run, generated the corresponding cluster histograms and peer data summaries, spread these peer data summaries using a `PROBE/ECHO` run and then ran 100 test queries for that configuration.

In table 1 we present the communication cost of global clustering. Here, we clustered the image collection into 12, 25, 50, 100 clusters. As we said, the number of cluster centers that has to be passed on in a clustering iteration is proportional to the number of clusters, it is also proportional to the number of edges. The same, it influences the cost of each peer representation. The total number of bytes of peer representations that have to be held in each peer is shown in the fourth column. That is, each peer spends more communication cost on creating the clustering than on obtaining all cluster representations. That is in small networks, the cost is more strongly influenced by the frequence of clustering runs, than by obtaining peer representations.

#clusters	#iterations	#bytes per run and peer	total #bytes of all peer representations
12	20	72000	41760
25	20	150000	79200
50	20	300000	151200
100	20	600000	295200

Table 1. Communication complexity of global clustering, and cost of peer representation, depending on number of global clusters.

We then varied the number of centers of interest per peer, as well as the number of global clusters to obtain Fig. 1 (left): for 2 (and 5, respectively) clusters of interest per peer, as well as 12, 25, 50, and 100 global clusters we show the median peer rank (as described in section 4.3). Here we see that with an increasing number of global clusters, our approximation gets more precise, yielding better performance. Performance varies from twice as good as the baseline (visiting $0.2 \cdot N_{Peers}$ before finding 50% of the top 20 matches) to about 10 times as good

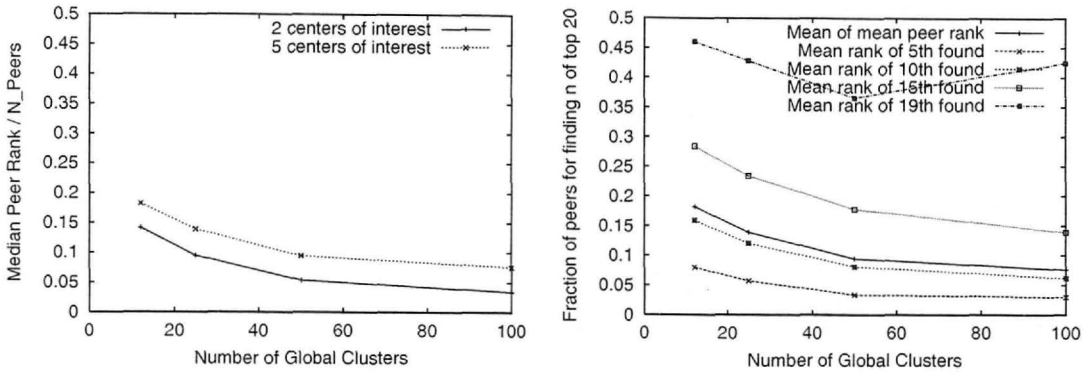

Figure 1. Left: Median peer rank depending on the number of global clusters for two numbers of (local) clusters of interest. **Right:** For 5 centers of interest and 12 to 100 global clusters, we show how many peers it takes on average to find 5, 10 (*i.e.* median), 15 and 19 documents of the top 20. We also plot the average mean rank of the peers in which documents of the top 20 were found.

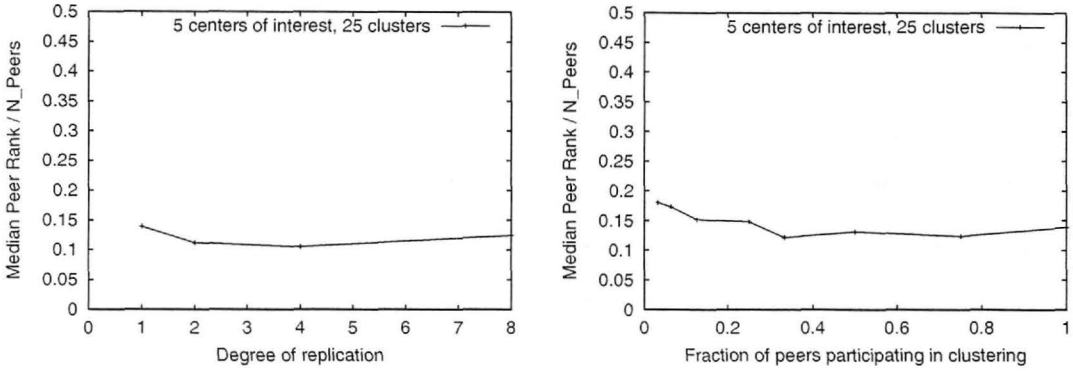

Figure 2. Left: Median peer rank depending on the degree of replication. **Right:** Median peer rank, if only a fraction of the peers has participated in the clustering.

$(0.04 \cdot N_{Peers})$. Fig. 1 (right) shows modifications of the median peer rank measure: the 25% quantile (5 images out of the top 20), the mean, as well as the 75% quantile and the 95% quantile. We show the 95% quantile, as it shows that for the 19th match, our cluster histogram peer representation exhibits overfitting, if there are too many global clusters: the results get worse.

Fig. 2 (left) shows that replication (the degree of replication is varying from 1 to 8) is influencing only weakly the median peer rank.

Finally Fig. 2 (right) shows how the quality of the clustering influences the quality of query results. To this end, we deliberately degraded the clustering quality by using *only a fraction of the total network*, thus only a *fraction (varying from* $\frac{1}{1}$ *to* $\frac{1}{8}$*) of the total data collection in the network* for calculating the global clustering. We can see from the results in this figure, that it suffices to do a global clustering of only a part of the collection for obtaining good results. This means, that even in a P2P system used "in the wild" with joins and leaves, clustering would have to be performed fairly rarely for maintaining good retrieval performance.

6. CONCLUSION

In P2P networks for CBIR, peer data summaries can be used within the retrieval process for increasing efficiency. Within this article we described new experiments evaluating the usefulness of cluster histograms used as peer data summaries.

The results are very encouraging: retrieval itself is efficient, and our measurements suggest that costs for maintaining the peer data summaries are low, because the most costly step of maintaining peer data summaries, the clustering has to be performed only rarely.

Future work comprises the following main areas:

- Peer data representations and feature sets: it will be interesting to use other data representations, such as gaussian mixture models, and other feature sets, such as texture features.
- Unreliable environments: In the current experiments, no peer participating in a clustering left the network during the clustering. While our current results suggest, that our method will be not very susceptible against such failures, we would like to propose measures for unreliable networks to generate useful clusterings.
- The scenario itself: Just as in CBIR, there is a need for suitable benchmarking scenarios with test data and ground truth. As we have made clear in this paper, in the case of P2P CBIR, the benchmarking scenario needs to comprise suitable assumptions about the data distribution over peers.

REFERENCES

1. Gregory R. Andrews. Paradigms for process interaction in distributed programs. *ACM Computing Surveys (CSUR)*, 23(1):49–90, 1991.
2. Clip2. The Gnutella Protocol Specification v0.4. `http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf`, 2000.
3. Francisco Matias Cuenca-Acuna and Thu D. Nguyen. Text-Based Content Search and Retrieval in ad hoc P2P Communities. Technical Report DCS-TR-483, Department of Computer Science, Rutgers University, April 2002. PlanetP.
4. Inderjit S. Dhillon and Dharmendra S. Modha. A data-clustering algorithm on distributed memory multiprocessors. In *Large-Scale Parallel Data Mining, Workshop on Large-Scale Parallel KDD Systems, SIGKDD, August 15, 1999, San Diego, CA, USA, revised papers*, volume 1759 of *Lecture Notes in Computer Science*, pages 245–260. Springer, 2000.
5. Martin Eisenhardt, Wolfgang Müller, and Andreas Henrich. Classifying documents by distributed P2P clustering. In Springer, editor, *33. German National Conference on Computer Science* (GI Jahrestagung) *Workshop Web Information Retrieval*, LNI, November 2003.
6. Amr Z. Kronfol. A Fault-tolerant, Adaptive, Scalable, Distributed Search Engine. URL: `http://freenetproject.org/cgi-bin/twiki/view/Main/FASD`, May 2002. Final Thesis, Princeton.
7. Jinyang Li, Boon Thau Loo, Joe Hellerstein, Frans Kaashoek, David R. Karger, and Robert Morris. On the feasibility of peer-to-peer web indexing and search. In *2nd International Workshop on Peer-to-Peer Systems*, Berkeley, CA, February 2003.
8. Wolfgang Müller and Andreas Henrich. Fast Retrieval of High-Dimensional Feature Vectors in P2P Networks Using Compact Peer Data Summaries. In *electronic Proceedings of ACM MIR'03 Workshop*, Berkeley, California, USA, November 2003.
9. The neurogrid site. `http://www.neurogrid.net/php/index.php`.
10. Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In *Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications*, pages 161–172. ACM Press, 2001.
11. John Smith. *Integrated Spacial and Feature Image Systems: Retrieval, Compression and Analysis.* PhD thesis, Graduate School of Arts and Sciences, Columbia University, 2960 Broadway, New York, NY, USA, 1997.
12. Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, 2001.
13. Chunqiang Tang, Zhichen Xu, and Mallik Mahalingam. psearch: Information retrieval in structured overlays. In *First Workshop on Hot Topics in Networks (HotNets-I)*, Princeton, NJ, October 2002.
14. Nuno Vasconcelos. *Bayesian Models for Visual Information Retrieval.* PhD thesis, MIT, June 2000.