Clustering wavelets to speed-up data dissemination in structured P2P MANETs

Mihai Lupu[†] Jianzhong Li[‡] Beng Chin Ooi[† §] Shengfei Shi[‡ *]

[†]Singapore-MIT Alliance
National University of Singapore
{mihailup, ooibc}@comp.nus.edu.sg

[‡]School of Computer Science and Technology
Harbin Institute of Technology
{lijzh, shengfei}@hit.edu.cn

Abstract

This paper introduces a fast data dissemination method for structured peer-to-peer networks. The work is motivated on one side by the increase in non-volatile memory available on mobile devices and, on the other side, by observed behavioral patterns of the users. We envision a scenario where users come together for short periods of time (e.g. public transport, conference sessions) and wish to be able to share large collections of data. With hundreds and even thousands of data items stored on small devices, content publication is simply too energy and time consuming. By indexing summary information obtained by a combination of multi-resolution analysis and k-means, our method (Hyper-M) is able to cut down the overall construction time of an overlay network such as CAN by an order of magnitude, as well as provide fast approximate similarity search on such a network. The results of our extensive experimental studies confirm that Hyper-M is both energy and time efficient, and provides good precision and recall.

1. Introduction

Recently, all major mobile phone producers have introduced hand-held devices that boast gigabytes of storage capacity, as well as medium range communication facilities (bluetooth). Such small, portable devices are capable of storing hundreds of songs or photos, and tens of video files.

Given these facilities, one can imagine a scenario where a group of co-workers, who share a confined space for about eight hours per day, use their cellphones or mobile devices to search and exchange files among themselves. Even more familiar is a conference scenario where researchers come together to share information. Data sharing goes hand in hand with information sharing - the main purpose of any conference.

In practice, it is common to see groups being formed $ad-hoc$ly for relatively short periods of time (one to a few hours), during which there is limited mobility (e.g. in the office, in school, on long-distance public transport). For such data-sharing network to be useful, we have two contradictory objectives to meet. Firstly, due to the short duration of activities, the speed of deployment should be high. Secondly, while the retrieval could be approximate, the quality of retrieval should remain high as well. To the best of our knowledge, this is the first attempt to address such a problem.

Structured overlays have been proved to be efficient, but even the best of them take $O(\log N)$ for each insertion. This may be too long if each peer has to publish hundreds of items in only a few seconds. As a consequence, the only possible solution is not to publish all the data items. Instead, a summarization method should be used to derive representative summaries that describe the entire dataset of a peer. Clustering allows us to achieve just that. It also reduces data transfer across the network during insertion, saving bandwidth and enforcing copyright restrictions. However, clustering methods show a rapid decay in performance when the data is very high dimensional. On the other hand, multimedia data are often represented as very high dimensional feature vectors, such as histograms of colors [21] or tones [26]. We therefore need both clustering methods and dimensionality reduction techniques. While there have been a number of such proposals [3, 10, 22], none of them maps well to a distributed environment as they have to take into account all the data points.

In this paper, we propose a novel approach called Hyper-M[1] for speedy data indexing on an existing peer-to-peer structured network such as CAN [20]. Hyper-M's combination of k-means clustering and Discrete Wavelet Transform (DWT) enables publishing with low setup overhead and latency and good approximate retrieval.

The indexing of clusters, instead of individual data items, naturally and substantially reduces the index construction time and even though Hyper-M does not index individual

[‡]Partially supported by the Key Program of the NNSF of China under Grant No. 60533110.

[*]Work done while intern at School of Computing, NUS

[§]Partially supported by the SpADE Project

[1]We coined the term *Hyper-M* to symbolize the hyperspace representation of data in a MANET. It also induces the idea of speed.

data items in each peer, it provides efficient support for point, similarity range and k-nearest neighbor searches because it is able to exploit geometrical properties of its data and query representation. As we will show, our proposed approach offers good retrieval of up to 90% in terms of precision and recall with a tenfold increase in the speed of data insertion. As a side-effect of the DWT approach, we have observed that data is more equally dispersed in the network, thus alleviating the need for an additional load balancing mechanism. We make the following main contributions:

1. A fast data dissemination method based on a novel combination of clustering and wavelet transform, that works independently of the underlying overlay structure (Section 3). We include:
    - A theoretical analysis of cluster behavior under the wavelet transform;
    - A peer scoring method to estimate the likelihood that a particular peer holds query answers;
2. Query processing methods for the given framework (Section 4):
    - Theoretical proofs to guarantee accuracy and lack of false dismissals for range queries;
    - A heuristic method to answer k-nn queries;
3. Extensive experimental results regarding the speed of data dissemination (Sections 5) and the retrieval effectiveness (Section 6)
    - We also show that a better balancing of data distribution in the p2p overlay is achieved due to the orthogonality of vector spaces in the DWT.

The next section presents a summary of related work.

2. Background and Related Work

Hyper-M is part of a larger, comprehensive project under development at the School of Computing, NUS, entitled BESTPEER [1]. This framework provides a broad range of querying and searching facilities over various types of data sources and is built on a flexible P2P platform that can switch smoothly between structured and unstructured overlay.

2.1. Rapid P2P Deployment

A few recent studies have examined the problem of fast construction of overlay networks [2, 5]. The common approach is to parallelize network construction by initiating insertion simultaneously, assuming that a fair number of nodes are joining the network at the same time. Such works assume the existence of an initial random connection graph among the nodes and proceed by pairing them to form a tree structure. Aberer et al [2] define a top-down trie building process where peers randomly interact with each other and make parallel decisions on which regions of the data key space each should maintain, possibly splitting a region. Conversely, a bottom-up approach is taken in [5], where the tree structure is built by representing subtrees that are already formed as virtual peers and coupling them to form larger subtrees.

However, the main factor in the creation time of a useful peer-to-peer network is, at least in our opinion, the time of data dissemination. It is commonly expected that the number of data items largely exceeds the number of nodes. Thus, our approach complements the parallelization method, employing clustering and dimensionality reduction to achieve data summarization for a speedy data dissemination. In the next subsections, we review background work in clustering and dimensionality reduction, with particular attention to their potential use in the distributed environment.

2.2. Clustering Methods

For our system, we are interested in clustering methods that work in vector space models and provide simple cluster representations that can be used as a summary of the original data. Given this prerequisite, we choose the k-means algorithm whose results can be expressed as spheres in a vector space. K-means is a popular clustering method due to its invariance with respect to translations and orthogonal transformations of data points and robustness to outliers. Also, it does not depend on the order of appearance of the data items.

A good example of usage of k-means for indexing high-dimensional data is [23]. The work there is complementary to our current work in the sense that it illustrates an example of the type of information that could be inserted in the Hyper-M framework.

Various extensions of the basic k-means approach have been introduced [18, 19] mainly to speed up the process, but all the methods share a main problem of the basic approach: the decay of distance comparison efficiency in very high dimensional spaces. It has been shown [14] that for L_i-norms the difference between the closest and the furthest away point to a query approaches zero if $i > 2$ or a constant if $i = 2$, as dimensionality increases. Consequently, dimensionality reduction techniques are vital for optimal performance in the context of vector space models.

2.3. Dimensionality Reduction Techniques

In this section we review the techniques that achieve dimensionality reduction while maintaining enough information to allow a meaningfull clustering process. That excludes data signature methods such as Bloom filters [6] because, first, they do not maintain locality (except for Locality-Sensitive Hashing [8]), and second, more importantly, the clusters that might be obtained give no information about the appartenance of the original data items, because the hash functions used are not reversible.

Another way to reduce the dimensionality of the data is to consider only some of the features [10], but if the selec-

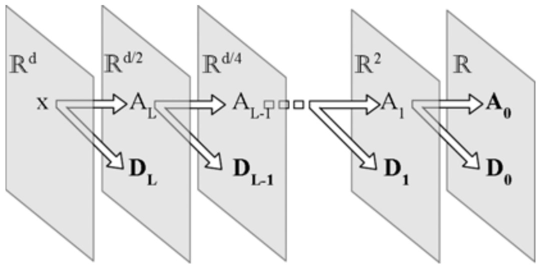


Figure 1. DWT schema

tion is done independently at each peer, the features selected may be different, making comparisons meaningless. A simpler method would be to just group together neighboring features [4, 22], but this method is too rigid and the density of the grid needs to be considered on a case by case basis.

Another category of dimensionality reduction methods consists of 'space projections'. Here, Singular Value Decomposition [12] is a linear algebra method that is argued to uncover latent semantic information present in the data. However, it incurs a very high computing cost ($O(m^2n)$ for a $m \times n$ matrix) and it relies on global information, which is not easy to collect in p2p networks. Similar problems occur with Principal Component Analysis [17]. It tries to identify key directions in the data such that differences among documents are emphasized. Mathematically, this translates into a projection onto a space whose basis vectors are chosen based on the variance of the data. A similar projection is obtained in FastMap [11], but in steps.

The Discrete Wavelet Transform (DWT) [9] is a signal analysis method that has been imported from the engineering realm. It owes its success to its linear computation time and the fact that it provides information not only about which frequencies are present in a signal (as Fourier analysis does), but also about when these frequencies occur. It has been shown to perform well in image compression and analysis, being able to provide good approximations of the original data as well as identify *interesting* features of the original image. It can also be applied in the distributed context because the vector spaces onto which it projects the data are independent of the data itself.

DWT projects the high-dimensional vector into two wavelet spaces resulting in two new vectors: the *approximation* (A) and the *detail* (D), each having half the length of the original vector. The process recursively decomposes each approximation (Figure 1) until the length of the vector is 1. Finally, the original vector can be restored by composing the last approximation (A_0, which in the rest of the paper will be denoted simply by A) and the series of details D_0, D_1, ... , D_L. The maximum number of levels is $\log_2(d)$, where d is the dimensionality of the data. The dimensionality of the data at each level l is 2^l.

Table 1. Notations

N	number of nodes
n	total number of data items
d	dimensionality of the dataset
K_p	number of clusters on a node p
x_i/q_i	an item in a data vector / query vector
α, β	angles (see Figure 4)
r/ϵ	radius of a data cluster / range query
l	a level of the wavelet hierarchy
A, D_l	levels of the wavelet hierarchy
L	number of wavelet levels used
$sphere_{c/q}$	sphere representing a data cluster / range query
$items_c$	number of data items in cluster c
C^l	a cluster in the l vector space
$Overlay_l$	p2p overlay managing data on the l^{th} level

3. Hyper-M

The *Hyper-M* framework provides a method to rapidly spread high-dimensional data in a MANET. The problem is not trivial because we need to summarize data such that 1. we maintain enough information to be able to provide good retrieval and 2. we do this without knowing the entire set of documents. Our solution involves both clustering and wavelet transform, as sketched in Figure 2. It works as follows: First, a peer decomposes all its data items using DWT, resulting in a series of $\log(d)$ vectors (step $i1$). This process could be done offline, and it does not add to the overall time complexity of the method. In fact, for image files, existing codecs already use the wavelet transform to compress data [25]. Subsequently, each subspace obtained by DWT is clustered independently (step $i2$), yielding a maximum of $\log(d)$ sets of clusters. (We note here that the reduced dimensionality at each level allows the formation of tighter clusters.) For each subspace we consider a separate p2p overlay, into which we insert these clusters (step $i3$). The number of subspaces used is a key factor in Hyper-M – our effectiveness experiments in Section 6 show that using more than four levels incurs additional overhead that is not justified by the improvements in precision and recall.

Despite its use of summaries, Hyper-M is able to maintain high effectiveness due to our proposed retrieval method, outlined in Figure 3. The search process works in two phases: first, based on the published summaries, it identifies which of the peers are most likely to contain the required data (steps $s1$ and $s2$ in Figure 3) and, second, it retrieves the actual data items from a subset of these peers (step $s3$). Each of these two phases has its own problems: 1. How to

```
Insertion(Peer $p$)
$i1$  for each item $d$ on $p$
          $\{x^l\}_{l\in\{A,D_0,\ldots,D_L\}}$=DWT($d$)
      for each level $l \in \{A, D_0, ..., D_L\}$
$i2$      $\{C^l_i\}_{0\le i<K_p}$=K_Means($l$)
$i3$      Overlay$_l$.insert($C^l_i$), $0 \le i < K_p$
```

Figure 2. Peer insertion in Hyper-M

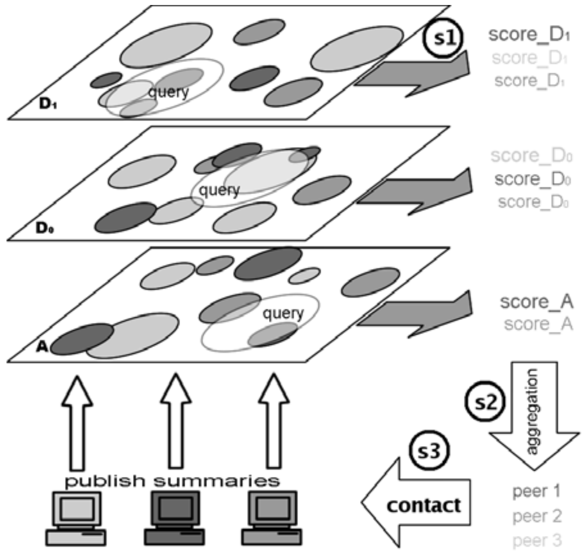


Figure 3. Hyper-M retrieval method

determine the potential amount of useful information that a peer holds and convert it into a score? 2. Based on these scores, how do we determine how many nodes we actually need to contact to retrieve the required amount of information? We answer the first question in Section 3.2, while to the second question we dedicate the entire Section 4, due to its importance for the quality of retrieval. Before that, let us briefly describe the nature of the clusters.

3.1. Cluster Representation

To reduce the complexity in representation and computation, clusters are represented as spheres in a vector space, as in [23]. Each representative cluster is described by a centroid and a radius, along with a count of the data items in the cluster. The count is used for estimating the relevance of a peer with respect to a query.

The representation of clusters as hyperspheres requires us to study how these hyperspheres behave under wavelet approximation. In particular, we want to know where the approximations of the original points appear in the reduced space.

Theorem 3.1. *All the points inside a sphere of radius r in the original vector space will be mapped inside a sphere of radius $\frac{r}{\sqrt{2^{\log d - l}}}$ in the level l approximation (or detail) space.*

Proof. For simplicity, we consider a sphere of radius 1 centered at the origin of the vector space. The same calculations are valid for any sphere, plus the necessary scaling and translations. All the points inside a d-sphere are given by the following parametric equations:

$$x_1 = \prod_{i=1}^{d-1} \cos \alpha_i$$

$$\dots$$

$$x_k = \sin \alpha_{k-1} \prod_{i=k}^{d-1} \cos \alpha_i, 2 \le k < d$$

$$\dots$$

$$x_d = \sin \alpha_{d-1}$$

where the angles $\alpha_i, i \in \{1, ..., d-1\}$ range between 0 and 2π to define all the points on the sphere.

When using the Haar wavelet[2], pairs of coordinates are taken and averaged. To build a hypersphere in the approximated space, the center is given by the approximation of the center of the original hypersphere, but the radius is likely to be smaller. To derive a formula for the new radius, we need to find what points in the original sphere will form the boundary of the new sphere. We thus need to find the maximum value on each new coordinate axis. Given the formulas of the Haar wavelet, we need to maximize the sum between two consecutive coordinates in the original space:

$$\max_{\alpha_{k-1} \dots \alpha_{d-1}} \left(\sin(\alpha_{k-1}) \prod_{i=k}^{d-1} \cos(\alpha_i) + \sin(\alpha_k) \prod_{i=k+1}^{d-1} \cos(\alpha_i) \right)$$

$$= \max_{\alpha_{k-1} \dots \alpha_{d-1}} \prod_{i=k+1}^{d-1} \cos(\alpha_i)(\cos(\alpha_k)\sin(\alpha_{k-1}) + \sin(\alpha_k))$$

From the formulas, we can see that the maximization problem is reduced to:

$$\max_{\alpha_{k-1}, \alpha_k} (\cos(\alpha_k)\sin(\alpha_{k-1}) + \sin(\alpha_k))$$

This function takes the maximum value of $\sqrt{2}$ in two points: $(\frac{\pi}{4}, \frac{\pi}{2})$ and $(\frac{3\pi}{4}, \frac{3\pi}{2})$. This, combined with the fact that we are using averages (the sum divided by two), gives us a new maximum distance from the center of $r/\sqrt{2}$.

Similarly, the same bound can be shown to exist for the first level of detail ($D_{\log d}$), and generalizing, for the l^{th} level of detail (D_l), the radius is $r/\sqrt{2^{\log d - l}}$. $\square$

3.2. Peer Relevance Score

The peer relevance score is a measure of the likelihood that a peer contains relevant information. This likelihood is computed based on an intersection of spheres, as illustrated in Figure 3. The formula is given by Equation 1.

$$Score_l = \sum_{c=0}^{K_p} \frac{Vol_{sphere_c \cap sphere_q}}{Vol_{sphere_c}} items_c \qquad (1)$$

where K_p is the number of clusters on peer p.

[2]We consider the Haar wavelet due to ease of proof. Similar, though more laborious proofs, can be done for other wavelets

The score is indexed by l because we have a different score at each detail level of our system. These values need to be aggregated into a global score used to chose the nodes that need to be contacted directly in order to retrieve the data. Throughout the experiments shown in this paper, we have used *the minimum score* approach:$Score = \min\{Score_A, Score_{D_0}, Score_{D_1}, ...\}$. It has the desirable property of pruning many candidate peers, thus reducing the amount of bandwidth required. For range queries, we show in Section 4.1 that this policy yields no false dismissals.

4. Query Processing

Quick publishing over a MANET is only useful if various similarity queries could be answered efficiently and with high accuracy. In Hyper-M, apart from point query, two similarity queries are supported: 1. Range queries: *retrieve all the items within an ϵ radius* and 2. k-nearest neighbor (k-nn) queries: *retrieve the k closest items*. Due to the transformations adopted in publishing, we have to ensure that queries can be answered with no false negatives. In general, Hyper-M's queries are resolved in two steps. First, assessing the relevance of each peer involves the translation of the query in the approximation or detail space and the aggregation of a peer relevance score based on cluster intersections. Second, given the scores of each peer, we must determine relevant nodes to answer the query and then extract the minimum amount of information from each of these nodes.

Point queries are straight forward. Due to space constraints, we shall focus on the more interesting range and k-nn queries in the next two subsections.

4.1. Range queries

Range queries are easier in our context because of the relevant mathematical proofs that can be derived for the behavior of such queries in the wavelet space. In this section, we show that given Theorem 3.1 and our minimum-score policy, no false dismissals can occur. Previous studies [7] show that using only a number of detail levels when comparing data items results in no false dismissal. Here, we complement that by showing that considering only those items that on each level are closer than a particular threshold yields no false dismissals.

Theorem 4.1. *A point x that in every approximation or detail space is within a distance $\frac{R}{\sqrt{2^{\log d - l}}}$ of another point q (distance from Theorem 3.1), is within distance $R\sqrt{(log d + 1)}$ of the same point q in the original vector space.*

Proof. Formally, we know that each returned result $x \in V_l$ has the property that $dist(x,q) < \frac{\epsilon + r}{2^{\frac{\log d - l}{2}}}$, where V_l is the vector space corresponding to the l level of detail, ϵ is the radius of the query sphere and r is the radius of the data sphere whose center is x. In what follows, $R = \epsilon + r$.

If a particular item x is in the final answer, then its approximation and details are all within their respective thresholds in their corresponding vector spaces. The most restrictive case is where we consider all levels of detail. To avoid a plethora of indices, we present here a vector with a dimensionality of 4. Its decomposition has one approximation in $\mathbb{R}$, a level of detail in $\mathbb{R}$ and one in $\mathbb{R}^2$.

Equations 2, 3 and 4 show the conditions that must be satisfied for a particular cluster to be retrieved.

$$\left(\frac{x_1+x_2+x_3+x_4}{4} - \frac{q_1+q_2+q_3+q_4}{4}\right) < \frac{R^2}{2^2} \quad (2)$$

$$\left(\frac{x_1+x_2-x_3-x_4}{4} - \frac{q_1+q_2-q_3-q_4}{4}\right) < \frac{R^2}{2^2} \quad (3)$$

$$\left(\frac{x_1-x_2}{2} - \frac{q_1-q_2}{2}\right) + \left(\frac{x_3-x_4}{2} - \frac{q_3-q_4}{2}\right) < \frac{R^2}{2} \quad (4)$$

By simple transformations and summation of the three equations, we have that $\sum_{i=1}^{4}(x_i - q_i)^2 < 3R^2$. This states that the distance between the data point and the query in the original vector space is smaller than $R\sqrt{3}$ and in general $R\sqrt{\log d + 1}$. □

The above theorem guarantees no false dismissals. That is, for a range query expressed in its original space, the results returned by resolving the transformed query in the reduced spaces are in a larger sphere than the original one.

4.2. K-nn Queries

The use of summaries restricts our ability to determine exactly on which peers are the k-closest items. Instead, in this section we provide a heuristic to estimate the radius of a range query that will retrieve the required number of items. The results of this heuristic are presented in Section 6.

Intuitively, given the sphere and the density, we can estimate the number of items that are retrieved by a range query. This number is a monotonically increasing function of the radius of the query. In our case we need the inverse function: *given the number of items, what is the range query that we need to use in order to retrieve this number of items?*. In what follows, we will first show how we define the direct function (from range value to number of items retrieved) and then use a numerical method to get the inverse.

To estimate the number of data items that may be retrieved by a query of a given radius ϵ, we need to find the proportion between the volume of the intersection of two hyperspheres (query and data cluster) and the volume of one of them (the data cluster). Considering the volume formulas in [23], in the case of a vector space whose dimensionality

$$\frac{Vol_{intersect}}{Vol_{sphere}}=1-\frac{1}{\pi}\left(\arccos\left(\frac{r^2+\epsilon^2-b^2}{2r\epsilon}\right)-\sum_{i=0}^{\frac{d-2}{2}}\frac{2^{2i}(i!)^2}{(2i+1)!}\left(1-\left(\frac{b^2+r^2-\epsilon^2}{4b^2r^2}\right)^2\right)^{\frac{2i+1}{2}}\left(\frac{b^2-\epsilon^2+r^2}{2br}+\frac{b^2+\epsilon^2-r^2}{2b\epsilon}\left(\frac{r}{\epsilon}\right)^{2i+1}\right)\right) \quad (7)$$

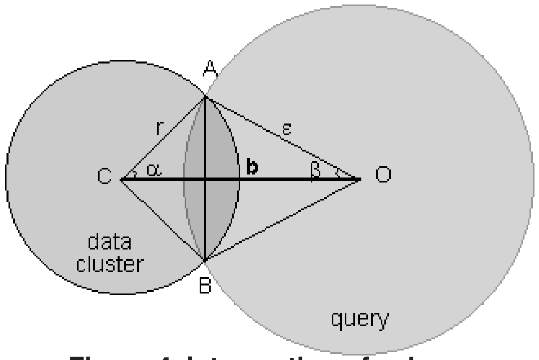


Figure 4. Intersection of spheres

is an even number, we derive:

$$\frac{Vol_{cap}}{Vol_{sphere}}=\frac{1}{\pi}\left(\alpha-\cos(\alpha)\sum_{i=0}^{\frac{d-2}{2}}\frac{2^{2i}(i!)^2}{(2i+1)!}(\sin(\alpha))^{2i+1}\right) \quad (5)$$

where α is the half-angle at the center, as depicted in Figure 4.

For the case where d is an odd number, the formulas are slightly different, but we do not present them here due to space constraints. The intersection of two spheres is the sum of two caps, so we have:

$$\frac{Vol_{intersect}}{Vol_{sphere}}=\frac{1}{\pi}\left(\alpha-\cos(\alpha)\sum_{i=0}^{\frac{d-2}{2}}\frac{2^{2i}(i!)^2}{(2i+1)!}(\sin(\alpha))^{2i+1}\right) + \frac{1}{\pi}\left(\beta-\cos(\beta)\sum_{i=0}^{\frac{d-2}{2}}\frac{2^{2i}(i!)^2}{(2i+1)!}(\sin(\beta))^{2i+1}\right) \quad (6)$$

Using the cosine rule, we can transform the angles α and β into functions of the radii of the two spheres and distance between their centers, as shown in Equation 7.

To obtain ϵ, we have to solve Equation 8, where the volumes are interpreted as functions of ϵ, and $\#clusters$ is the number of all reachable[3] clusters. Introducing Equation 7 in Equation 8 results in a high-order (almost) polynomial function of ϵ that does not have an analytical solution. We can compute the value of ϵ using numerical methods (e.g., the Newton method) which gives an approximated result.

$$k=\sum_{c=1}^{\#clusters}\frac{Vol_{sphere_c \cap sphere_q}}{Vol_{sphere_c}}items_c \quad (8)$$

[3]we provide details on reachability in Section 5

```
retrieveKnn($k$:items required, $L$:levels used)
1   for $l$ = 1 to $L$ do
2       estimate $\epsilon$ from $k$ and Equations 10 and 8
        do range query in subspace l with radius $\epsilon$
3       $< peer_i, score_i >$=rangeQuery($l$,$\epsilon$)
4   $< peer_i, score_i >$=mergeReturnedResults()
    compute the sum of the top P scores
5   for $p$ = 1 to $P$ do
6       sum = sum + $score_p$
    request a proportional number of items
    from each peer
7   for $p$ = 1 to $P$ do
8       $no\_items_p = C \times k(score_p/sum)$
9       result.add(getFromPeer($p$,$no\_items_p$))
10  result.sort()
```

Figure 5. k-nearest neighbor retrieval method

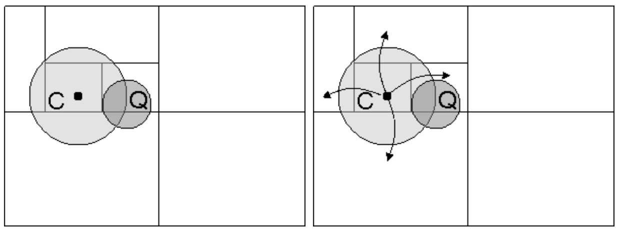
In a centralized environment, we would simply use this formula on the original dataset and obtain the necessary query radius to retrieve the top k items. In fact, the idea of obtaining a range based on a particular requirement of retrieved items has been analyzed before in optimization methods for k-nn queries [24], though in that work, the data areas are rectangular rather than circular as in our case. The main difference, however, is that in our context we perform this at the different approximation and detail levels of the wavelet representation and then merge the results. Our algorithm is outlined in Figure 5.

Steps 2 to 3 perform range queries on each level with different ϵ values. The intuition is to get from each level the k items required. Step 4 computes the number P of peers that need to be contacted.

The actual retrieval of data items is performed in Steps 5 to 10. In Step 8, the relative importance of a peer is computed as the ratio between its score and the sum of all the P relevant peers considered. The constant C is a tuning knob that can increase precision or recall, depending on application requirements (i.e. low bandwidth usage versus completeness of results).

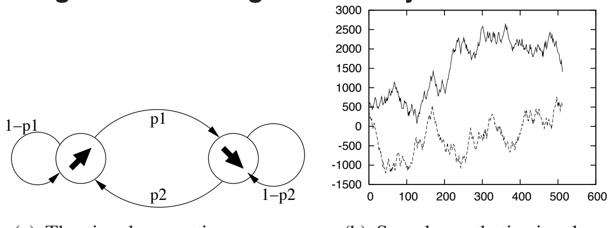
5. Experiments on Data Dissemination

Our method has been designed independent of the underlying peer-to-peer overlays, and it could be implemented on top of BATON[15], VBI-tree[16], CAN[20] or any peer-to-peer overlays for that matter so long as they can support multi-dimensional indexing. For demonstration and evaluation purpose, we use CAN as the underlying peer-to-peer overlay.



(a) Q cannot query C (b) replicate C in the regions it overlaps

Figure 6. Ensuring reachability of all clusters



(a) The signal generating process (b) Sample synthetic signals

Figure 7. Synthetic dataset

The insertion method is as described in the original CAN work [20]: the insertion point is given as a multi-dimensional point and the next step in the routing process is chosen to minimize the distance between the current location and the insertion point. In our case, the multi-dimensional vector used as an insertion point is the center of a cluster that needs to be published.

A problem specific to CAN when used to index non-zero sized objects is the possibility that the area of the object (i.e. cluster) overlaps more than one region. As depicted in Figure 6, the query Q would not retrieve the information present in data cluster C because the node its centroid belongs to does not have any information about that cluster. Replication cannot be avoided in this context, but we show that our method still provides a significant improvement in insertion time compared to the usual CAN approach without replication.

5.1. Synthetic Data

In order to test the speed of data dissemination we generated 100,000 512-dimensional feature vectors. To simulate real data, we used a Markov process with two states *Increasing* and *Decreasing*, as depicted in Figure 7a. The transition probabilities p_1, p_2 were generated randomly as follow: First, p_1 was chosen uniformly between 0 and 0.5. Then, $p_2 = p_1 + x$, where x was also chosen randomly between -0.05 and 0.05. The starting value, the initial state, the increase/decrease step, as well as the maximum step value were all chosen randomly. A sample of the resulting dataset is shown in Figure 7b. The data was subsequently clustered using k-means in the original vector space and then each cluster was redistributed among 8 to 10 nodes. This method simulates user behavior in the sense that each user commonly has a limited set of interests, thus maintaining items belonging to a subset of all the classes (i.e. clusters) in the data space. The number of clusters on each node is independent from other nodes, but in order to meaningfully test the effects of clustering on the insertion times and retrieval effectiveness, we have considered a fixed number of clusters on each node. Throughout the remaining of the section, efficiency tests were done in a 100-nodes network, each node holding 1000 items.

5.2. Speed of Insertion

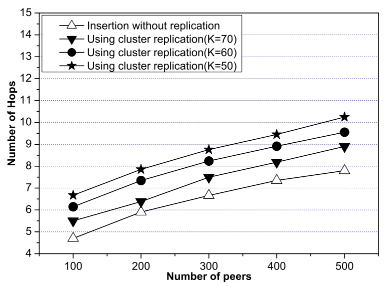
We implemented CAN in the Java programming language and simulated the parallel behavior of a peer-to-peer network with a scheduler class and an event queue. Every message generated in the network is sent to the event queue. Periodically, parallel execution is simulated by emptying the queue.

First we need to assess the impact of data replication on the speed of insertion. Figure 8a shows the average number of hops for different cluster sizes. As expected, if the clustering is finer, the number of hops approaches the no-replication standard. This happens because a smaller cluster has less chances of overlapping other zones than the one its centroid is located in.

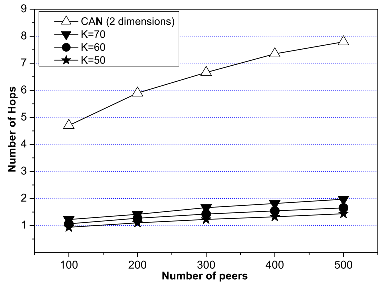
Our method not only overcomes this overhead, but provides up to 400% reduction in the number of hops compared with the basic CAN insertion method. Figure 8b shows the performance of insertion, when the amount of data inserted into CAN is considered. The tests clearly show that Hyper-M sets up the network overlay much faster, even if it incurs some replication overhead. In these tests, Hyper-M used four layers of network overlay. For illustration purposes, we implemented 2-dimensional CAN for the 512-dimensional dataset by indexing in only 2 dimensions. Though it cannot be used to retrieve meaningful data, it shows that Hyper-M, even with 4 levels of overlay, performs better than a 2-dimensional CAN that has to insert all its data items individually.

It is important to see how the system behaves when different number of overlays are used. Figure 8c shows the average number of insertion hops necessary to build the overlay(s). We see that Hyper-M greatly reduces the number of hops required to publish each item when compared to the CAN approach in the original vector space (512-dimensional). Again, in this figure, we plotted the insertion performance of 2-dimensional CAN to have a feel on the magnitude of the performance gap.

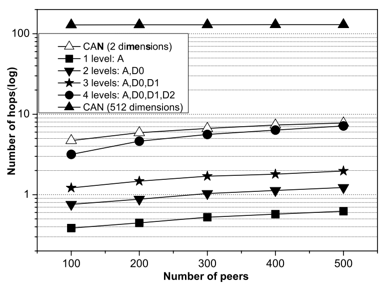
We note that some values for the average number of hops are smaller than 1 because we are averaging over the number of items on a peer, but insert only cluster centroids. This is the main point of our work - summarization allows us to reduce drastically the total number of hops for inserting all the data items, while maintaining sufficient information to allow for good precision and recall of retrieval, as demonstrated in Section 6.



(a) Cluster replication overhead



(b) Average number of hops per item insertion, function of the number of clusters on a peer



(c) Average number of hops per item insertion, function of the number of layers in the overlay (log scale)

Figure 8. Insertion speed tests

5.3. Data Distribution in the CAN Network

Our experiments in the previous section also pointed out an unexpected, beneficial side-effect of the wavelet transform. We refer here to the uniform data distribution among the nodes of the network: throughout our many experiments with different datasets, we have observed that data distribution in the CAN overlay tended to be fairly smooth, regardless of the data we used. Our assumption is that this is due to the orthogonality property among the different wavelet subspaces. This property formally guarantees the independence between the location of the different projections of the same vector. This means that different levels of the wavelet transform of the same data item are placed independently, thus achieving a better overall distribution.

To further prove this observation, we intentionally skew our data and observe its behavior in the wavelet subspaces. We cluster our original data and select only a fixed number of clusters (two to five in our experiments). We then apply the wavelet transform to the items in each cluster, and insert them into their respective overlays. Figure 9 shows the number of items on a peer in each of the possible overlays, as well as the average number of peers holding the data. As expected, the CAN overlay of the dimensionality of the original dataset performs among the worst, having most of the data on a very small number of nodes. The absolute worst case in terms of data distribution occurs with the usage of only the approximation level. This is to be expected because each wavelet space clusters the data better than the original space, thus putting the data only on very few nodes. However, as detail levels are added, the nodes used turn out to be from different parts of the overlay due to the orthogonality of the spaces.

The average number of data items on each peer provides a good indication that the wavelet spaces naturally distribute the data among the nodes. It is important to note that this is achieved without any explicit data redistribution.

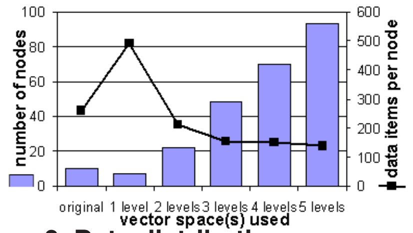


Figure 9. Data distribution among nodes

6. Effectiveness of Retrieval

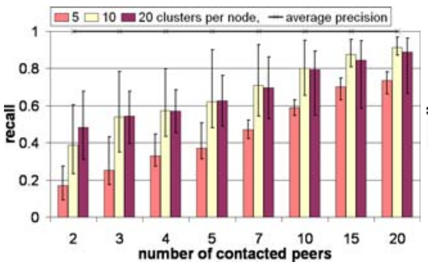
In this section, we present our experiments to evaluate the effectiveness of retrieval, using a real dataset [13] (The Amsterdam Library of Object Images). We use the standard *precision* and *recall* measures to evaluate the accuracy of our method. We implemented a centralized flat file system that indexes the data using the original vectors, and use the retrieval results as the basis for evaluating the effectiveness of our proposal. We show that Hyper-M achieves the necessary retrieval effectiveness, despite the fact that we only use summaries to represent the data items.

Throughout our experiments we also evaluate how the number of clusters on each peer influence the retrieval results. We expect that the finer the clusters, the higher the precision and recall because the data that is actually inserted into the network overlay is closer to the actual data.

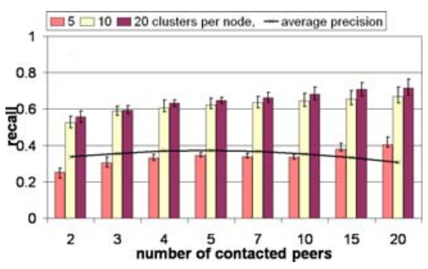
The subsequent figures plot averages of recall as vertical bars accompanied by error bounds showing the minimum and maximum value obtained in our experiments for some particular parameters. The variation is obtained by testing with different radii in the case of range queries, or different numbers (k) of requested items in the case of k-nn queries.

6.1. Retrieval Effectiveness on Real Data

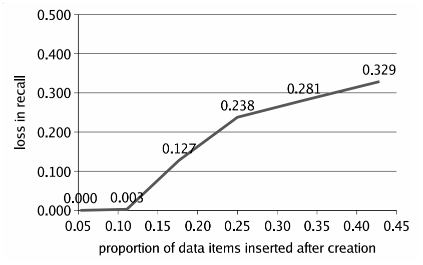
We tested our prototype system on an image dataset where each item was represented by a histogram of colors. The dataset contains 12,000 images representing a diverse set of objects under different angles and illuminations. Our purpose in this test is not to propose a new image retrieval method, but to assess the effectiveness of the image retrieval process when performed in a distributed environment using



(a) range queries



(b) k-nn queries



(c) insertions without update

Figure 10. Retrieval results

Hyper-M. In the following tests, we use a network of 50 nodes, each node containing, on average, a little over 200 image histograms.

Effectiveness of Range Queries. Range queries perform well thanks to the theoretical results in the previous sections. Precision is constantly 100% because once we decide which peers to contact, the query is performed directly on those peers, thus retrieving only the relevant items. It is more interesting to look at the recall of the method. Figure 10a shows that the recall reaches as high as 96% if enough peers are contacted. This is natural, as no system will ever be able to retrieve data without contacting the nodes that hold it.

Effectiveness of K-nn Queries. We test our method presented in Figure 5, in particular, the estimation of relevant items present on a peer, as calculated in Line 8 of this figure ($no\_items_p = C \times k(score_p/sum)$). The formula first normalizes the scores among the contacted peers and then requests from each of them a proportional amount of items. Figure 10b shows that the system performs well, balancing precision and recall at over 50%. As mentioned in Section 4.2, the constant C can be used to trade recall against precision and experiments (not shown here) confirm this expectation. It takes any positive value, but reasonable values are between 1 and 2 (i.e. demanding exactly the estimated number of documents or demanding the double). In brief, our experiments show that we obtain a 14.51% increase in recall when C is 1.5 (50% more data items retrieved) but also a drop of 21.05% in precision. Increasing C further to 2 adds an additional 4.23% to recall and substracts 6.67% from precision. It is thus, in general, more costly to have a higher value for C and it should be used only when completeness of results is very important.

Our expectation that a higher number of clusters on each peer improves the overall performance of the retrieval has been proved by the experiments. We note that using ten clusters instead of five almost doubles the performance, but using twenty instead of ten only increases it slightly. This shows that the performance is almost at its maximum even when using only ten clusters, which, considering the fact that we have around 200 items on each peer, represents a reduction by a factor of 20 - similar to the one used for our performance tests in Section 5.

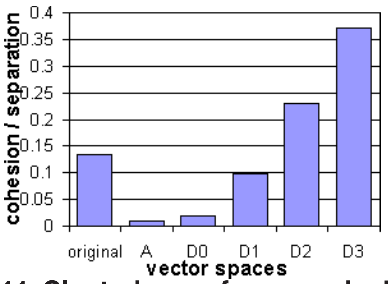


Figure 11. Clustering performance in different vector spaces

Finally, we note that our application scenario emphasizes the creation speed of the overlay. During the short life-time of the network, we expect that most new data items fit into the existing clusters. However, we have evaluated the impact of inserting documents after the creation of the overlay. Figure 10c shows the loss in recall versus the number of new documents inserted after the creation of the overlay. We can see that even if we insert as much as 45% new documents (3600 new data items, versus 8400 existing), the recall loses only up to 33%.

6.1.1. Effects of Multiresolution Decomposition Over Clustering Performance

After analyzing the results, we observe that one of the reasons for which Hyper-M is able to retrieve data reliably is the fact that the different levels of the multiresolution analysis allow for a better clustering, as measured by the proportion between *cohesion* and *separation*. Cohesion is the average distance of elements within the same cluster and separation measures the average distance between the centroids of different clusters. Thus, the proportion between them is a measure of the 'goodness' of the clusters. Figure 11 shows that the clusters created in the first three wavelet vector spaces are tighter and better separated than clusters created by the same algorithm in the original data space. Figure 11 also shows that as the level of detail increases, clustering stops performing as well. Certainly, this is mainly due to the dimensionality reduction present in the first levels of the wavelet transform, but also because those levels concentrate the summary of the information present in the original vectors. These results also motivated us to use only four wavelet levels when performing the effectiveness tests presented in the previous sections.

7. Conclusion

In this paper we studied the problem of fast index construction in an ad-hoc structured p2p network, with possible applications for short-lifespan mobile networks such as a MANET. We proposed a novel solution called Hyper-M for fast data dissemination. At each peer, it first transforms the data space using the discrete wavelet transform, and then applies a simple clustering method, k-means, on the transformed dataset to generate summaries that represent the dataset. It uses spheres to represent these summaries in the transformed space, and exploits geometrical properties in its range and k-nn search strategies. Our extensive experimental studies show that insertion cost per item drops by an order of magnitude compared with conventional approaches, while retrieval performance is as high as 90% in terms of precision and recall.

We have observed that applying the DWT before distributing the data results in a more homogeneous distribution across the network and conclude that this is due to the mathematical properties of this transformation. This alleviates the need for data balancing in the p2p network.

References

[1] Bestpeer: A one-platform multi-structures p2p system. http://xena1.ddns.comp.nus.edu.sg/p2p/.

[2] K. Aberer, A. Datta, M. Hauswirth, and R. Schmidt. Indexing data-oriented overlay networks. In *Proc. of VLDB Conf.*, 2005.

[3] C. Aggarwal and P. S. Yu. Finding generalized projected clusters in high dimensional spaces. In *Proc. of SIGMOD Conf.*, 2000.

[4] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proc. of SIGMOD Conf.*, 1998.

[5] D. Angluin, J. Aspnes, J. Chen, Y. Wu, and Y. Yin. Fast construction of overlay networks. In *Proc. of SPAA*, 2005.

[6] B. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of ACM*, 13(7), 1970.

[7] K.-P. Chan and A. Fu. Efficient time series matching by wavelets. In *Proc. of ICDE*, 1999.

[8] T. Darrell, P. Indyk, and G. Shakhnarovich, editors. *Locality-sensitive hashing using stable distributions*. MIT Press, to appear.

[9] I. Daubechies. *Ten lectures on wavelets*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1992.

[10] J. G. Dy and C. E. Brodley. Feature subset selection and order identification for unsupervised learning. In *Proc. of ICML*, 2000.

[11] C. Faloutsos and K.-I. Lin. FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proc. of SIGMOD Conf.*, 1995.

[12] G. W. Furnas, S. T. Dumais, T. K. Landauer, R. A. Harshman, L. A. Streeter, and K. E. Lochbaum. Information retrieval using singular value decomposition model of latent semantic structure. In *Proc. of SIGIR Conf.*, 1988.

[13] J. M. Geusebroek, G. J. Burghouts, and A. W. M. Smeulders. The Amsterdam library of object images. *Int. J. Comput. Vision*, 61(1), 2005.

[14] A. Hinneburg, C. Aggarwal, and D. Keim. What is the nearest neighbor in high dimensional spaces. In *Proc. of VLDB Conf.*, 2000.

[15] H. V. Jagadish, B. C. Ooi, and Q. H. Vu. Baton: A balanced tree structure for peer-to-peer networks. In *Proc. of VLDB Conf.*, 2005.

[16] H.V. Jagadish, B. C. Ooi, Q. H. Vu, Z. Rong, and A. Zhou. Vbi-tree: A peer-to-peer framework for supporting multi-dimensional indexing schemes. In *Proc. of ICDE*, 2006.

[17] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.

[18] L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data: an Introduction to Cluster Analysis*. John Wiley and Sons, 1990.

[19] R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *Proc. of VLDB Conf.*, 1994.

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

[21] H. Shawney and J. Hafner. Efficient color histogram indexing. In *Proc. of ICIP*, 1994.

[22] G. Sheikholeslami, S. Chatterjee, and A. Zhang. Wavecluster: A multi-resolution clustering approach for very large spatial databases. In *Proc. of VLDB Conf.*, 1998.

[23] H. T. Shen, B. C. Ooi, and X. Zhou. Towards effective indexing for very large video sequence database. In *Proc. of SIGMOD Conf.*, 2005.

[24] Y. Tao, J. Zhang, D. Papadias, and N. Mamoulis. An efficient cost model for optimization of nearest neighbor search for low and medium dimensional spaces. *IEEE TKDE*, 16(10), 2004.

[25] D. Taubman and M. Marcellin. *JPEG2000*. Kluwer International Series in Engineering and Computer Science, 2002.

[26] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE TSAP*, 10(5), 2002.