Query Workload-Aware Overlay Construction Using Histograms

Georgia Koloniari, Yannis Petrakis, Evaggelia Pitoura and Thodoris Tsotsos
Department of Computer Science, University of Ioannina, Greece.
{kgeorgia, pgiannis, pitoura, thodoris}@cs.uoi.gr

ABSTRACT

Peer-to-peer (p2p) systems offer an efficient means of data sharing among a dynamically changing set of a large number of autonomous nodes. Each node in a p2p system is connected with a small number of other nodes thus creating an overlay network of nodes. A query posed at a node is routed through the overlay network towards nodes hosting data items that satisfy it. In this paper, we consider building overlays that exploit the *query workload* so that nodes are clustered based on their results to a given query workload. The motivation is to create overlays where nodes that match a large number of similar queries are a few links apart. Query frequency is also taken into account so that popular queries have a greater effect on the formation of the overlay than unpopular ones. We focus on range selection queries and use histograms to estimate the query results of each node. Then, nodes are clustered based on the similarity of their histograms. To this end, we introduce a *workload-aware edit distance metric* between histograms that takes into account the query workload. Our experimental results show that workload-aware overlays increase the percentage of query results returned for a given number of nodes visited as compared to both random (i.e., unclustered) overlays and non workload-aware clustered overlays (i.e., overlays that cluster nodes based solely on the nodes' content).

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems – Distributed Databases, Query Processing; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – Clustering, Search Process; H.3.4 [**Information Storage and Retrieval**]: Systems and Software – Distributed Systems, Information Networks, Performance Evaluation

General Terms

Algorithms, Measurement, Performance, Experimentation

Keywords

Peer-to-Peer systems, Small Worlds, Range Queries, Overlay Network, Query Routing, Clustering

CIKM'05, October 31–November 5, 2005, Bremen, Germany.

1. INTRODUCTION

In peer-to-peer (p2p) computing, a large and dynamically changing set of autonomous computing nodes (the peers) cooperate to share resources and services. Since, it is not possible for a node to know about (i.e. connect with) all other nodes in the system, each node connects with a subset of the system nodes, thus forming an *overlay network* on top of the physical one, for instance, on top of the Internet. A query for data may be issued at any node. The query is routed through the overlay network from the issuing node towards nodes that have data satisfying it. We are interested in maximizing the number of query results returned for a given number k of nodes visited.

Ideally, for each query, we would like to visit only the best k nodes for the query, that is, the k nodes that have the most results. One way to attain a large number of results is by clustering nodes based on their content so that nodes with similar content are nearby in the overlay network (e.g., [2]). Then, once a query is routed to the appropriate cluster, relevant nodes are nearby. In this paper, we propose forming clusters not based solely on the content of the nodes but also on the *query workload*, so that both the type and the frequency of queries is taken into account in constructing the overlay.

In particular, we consider workload-aware overlays in the form of small-worlds. Intuitively, the topology of a small-world network represents a number of smaller networks (clusters) of relevant nodes that are rich in links between their nodes (short-range links), while they are linked to each other with a few random connections (long-range links) [24]. We define the relevance of two nodes based on their query results to the system workload. The reason for such "workload-aware" small-world overlays is that once in the appropriate group, the most relevant to a query nodes are a few short-range links apart. Long-range links are used for routing among groups.

We present an approach for building such overlays based on a decentralized procedure that exploits local indexes. A local index is a characterization of the content of a node. To attain workload-awareness, we define similarity among indexes, so that nodes with similar indexes match similar sets of queries. Similarity is also *weighted*, so that matching frequent queries counts "more" than matching infrequent ones.

To demonstrate the feasibility of workload-aware clustering, in this paper, we consider range queries. For instance, the attributes of the query may correspond to the characteristics of music files (such as release-year, artist-name) in a music file-sharing system or the available resources (such as CPU, memory) in a resource-sharing p2p system. We use histograms as local indexes to estimate the query results offered by each node. Furthermore, we propose a workload-aware edit-based distance metric between histograms that takes into account the workload (both the type and frequency of the queries).

Related Work. There are two basic types of p2p overlays: structured and unstructured ones. In *structured p2p* systems, data items (or indexes) are placed at specific nodes usually based on distributed hashing (DHTs) such as in CAN [18], Chord [10] and P-Grid [1]. In DHTs, each data item is associated with a key and each node is assigned a range of keys and thus items. Structured overlays usually follow a regular topology, such as a ring or grid. Recently, researchers have proposed extending DHTs (e.g., Chord) with long range links towards creating small-worlds [14]. Other extensions, instead of associating keys to data items based just on their identifier, associate with each data item (or node) a vector describing its content extracted using IR algorithms and then use this vector as input to the hash functions [21, 20]. However, this creates a dimensionality reduction problem, since the dimension of the vectors should match the dimension of the DHT. These proposals can collectively be seen as building content-based small-worlds in DHT-based p2p systems. In this case, the usual problems with structured p2p systems arise. The DHT topology is regulated since all nodes have the same number of neighbors and the selection of neighbors is strictly determined by the DHT semantics. Furthermore, sophisticated load balancing procedures are required and often content movement is necessary.

In *unstructured p2p* systems, there is no assumption about the placement of data items at the nodes. Many recent research efforts focus on clustering nodes based on the content in such non DHT-based overlays. In most cases, the number or the description of the clusters is fixed and global knowledge of this information is required. In [2], nodes are partitioned into topic segments based on their documents. A fixed set of clusters is assumed, each one corresponding to a topic segment. Clusters of nodes are formed in [22] based on predefined semantic categories of their documents. Similarly, [7] assumes predefined classification hierarchies based on which queries and documents are categorized. The clustering of nodes in [13] is based on the schemes of the nodes and on predefined policies provided by human experts. Associative overlays [5] cluster nodes into groups called guide rules, by examining whether their content satisfies some predicate. Our approach does not assume any knowledge of the clusters and extends content clustering by incorporating the query workload in the formation of clusters.

Our main focus in this paper is not on range query processing; we just consider range queries to demonstrate how the type of queries influences the creation of clusters. There has been a number of proposals for processing range queries in structured p2p systems [19, 9, 3]. A central issue is again dimensionality reduction.

Contributions. In summary, in this paper,

- we propose building *workload-aware* overlays where the grouping of similar nodes is not based solely on the content of the nodes but also on the query workload,
- we exploit the use of histograms as indexes for constructing workload-aware overlays and introduce appropriate histogram distance metrics that incorporate the frequency and type of queries.

Our experimental results show that our small-world construction procedure is effective, since in the resulting peer-to-peer system, the distance of two nodes in the overlay is proportional to the distance of their histograms. Furthermore, routing is very efficient, in particular, for a given number of visited nodes, there is an increase of about 60% in the number of results returned when compared to a non-clustered overlay and 40% when compared to a non workload-aware clustered one.

Paper organization. In Section 2, we motivate the need for workload-aware small-world overlays. In Section 3, we propose appropriate histogram distance metrics. In Section 4, we describe how histograms are used in routing and constructing workload-aware overlays. In Section 5, we present experimental results, while Section 6 concludes the paper.

2. WORKLOAD-AWARE OVERLAYS

We assume a p2p system with a set N of nodes n_i. The set N of nodes changes as new nodes leave and join the system. Each node is connected to a number of other nodes called its *neighbors* forming an overlay network of nodes. A query q for an item may be posed at any of the nodes, while items that satisfy the query may be stored at various nodes. The query is routed in the overlay from the node that posed the query to nodes offering items that match it. We are interested in maximizing the number of query results returned for a given number of nodes visited.

Ideally, we would like to route each query q only through those nodes that have the largest number of results for q. To this end, we define *PeerRecall*. Intuitively, *PeerRecall* expresses how far from the optimal a routing protocol performs. Let $results(n_i, q)$ be the number of results for query q returned by node n_i. Let V be the set of nodes ($V \subseteq N$), visited by a routing protocol, $Sresults(V, q)$ denotes the sum of the number of results for query q returned by each node that belongs to V.

$$Sresults(V, q) = \Sigma_{n_i \in V} results(n_i, q) \tag{1}$$

Definition 1 (PeerRecall). *Let $Visited$ ($Visited \subseteq N$) be the set of nodes visited during the routing of a query q and Optimal ($Optimal \subseteq N$) be the set of nodes such that $|Optimal| = |Visited|$ and $n_i \in Optimal \Leftrightarrow results(n_i, q) \geq results(n_j, q)$, $\forall\ n_j \notin Optimal$. We define $PeerRecall$ as: $PeerRecall(Visited, q) = \frac{Sresults(Visited,q)}{Sresults(Optimal,q)}$.*

To increase *PeerRecall*, for each query, the nodes offering the most results should be close to each other in the overlay network, thus forming clusters of relevant nodes. We are interested in creating such overlay topologies in the form of small-worlds. Formally, small-world networks are characterized by: (a) a small diameter and (b) a large clustering coefficient [24, 11]. The *diameter* of the network is the maximum network-distance between any two nodes in the network, where network-distance is the shortest path between the two nodes. The *clustering coefficient* captures the probability that two neighbors of a node are also neighbors themselves; it is the average fraction of pairs of neighbors of a node that are neighbors of each other. Intuitively, the topology of a small-world network represents a number of

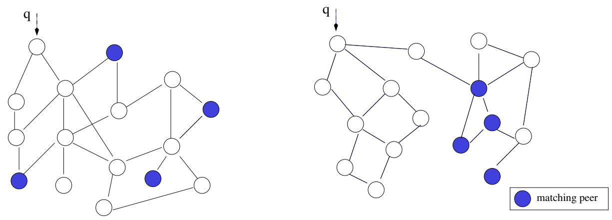

Figure 1: (left) ramdom and (right) small-world overlay.

smaller networks (groups or clusters) that are rich in links between their nodes (short-range links). These smaller networks are linked to each other through a few connections (long-range links).

Figure 1 shows such a small-world overlay network and a random (non clustered) one. In a workload-aware small-world network, nodes that match a query q are a few short-range links away, thus once in the right group all matching nodes are nearby. Long-range links are used to locate the right group. To build such small-world overlays, we consider both the content of each node and the query workload.

Definition 2 (Query Workload). *A query workload is a set W of pairs (q_m, f_m) where q_m denotes a query and f_m its frequency.*

For example, consider keyword-value queries and the workload W = { (x='A', 3/4), (x='B', 1/4)}. Consider three nodes n_1, n_2 and n_3. Assume that node n_1 has data items with values 'A' and 'B', node n_2 with values 'A' and 'C' and node n_3 with values 'B' and 'D'. Most probably a non workload-aware clustering approach would consider nodes n_2 and n_3 equally similar to node n_1, since both share one common value with it. However, when taking into account the workload, node n_2 is more "similar" to n_1, since n_1 and n_2 match a larger part of the workload than n_1 and n_3.

To construct workload-aware overlays, each node maintains a summary of the items stored locally; this is called a *local index*. Let $LI(n_i)$ denote the local index and $S(n_i)$ the number of items of node n_i. We create clusters of nodes that have *similar local indexes* taking into account a given workload. To this end, the distance (d) between the local indexes of two nodes must be descriptive of "how much" the two nodes match a given workload. In particular, than)

Property 1 (workload-awareness property). *A distance metric d between two local indexes is said to be* workload-aware *for a workload W, if for any three nodes n_1, n_2 and n_3,*

$$\frac{d(LI(n_1), LI(n_2))}{d(LI(n_1), LI(n_3))} \propto \frac{\Sigma_{q_m \in W} f_m |results(n_1, q_m) - results(n_2, q_m)|}{\Sigma_{q_m \in W} f_m |results(n_1, q_m) - results(n_3, q_m)|}$$

To estimate the number of results ($results(n_i, q)$) provided by each node n_i, we use histograms as local indexes.

3. USING HISTOGRAMS

Histograms are widely used to approximate data distributions [12]. We have previously used histograms to route queries in p2p systems [15]. Here, we use them to construct workload-aware overlays. First, we focus on retrieving data items (tuples) from a relation $\mathcal{R}$ (set of tuples) based on a single attribute x. We then extend our approach for selections on more than one attribute. Histograms can be also used to estimate the results of more complex queries (for instance, queries involving joins); our approach can be readily extended for such cases. As usual, we use the notation $t.x$ to denote the value of attribute x of a tuple $t \in \mathcal{R}$.

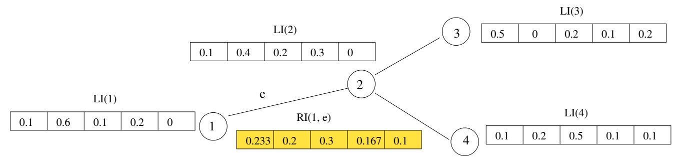

Figure 2: The local indexes of nodes 1, 2, 3, and 4 and the routing index of link e of node 1 for radius $R = 2$, assuming that local indexes $LI(2)$, $LI(3)$ and $LI(4)$ have the same size and b (the number of buckets) is 5.

3.1 Histograms as Routing Indexes

Building accurate histograms is a well-studied problem (e.g., [17, 12]). Since, our focus is on showing how histograms can be used to construct workload-aware overlays, we consider simple *equi-width* histograms. The *equi-width* histogram H of an attribute x is an array H_i constructed as follows. Without loss of generality, assume a numerical domain for attribute x, $\mathcal{D} = \{0, \mathcal{M} - 1\}$. The domain of attribute x is divided into b mutually disjoint ranges of equal width r called *buckets*. For each such bucket i, $0 \le i \le b - 1$ of H, H_i is the average frequency of values in bucket i, that is, the number of tuples (data items) for which it holds $i\,r \le t.x < (i+1)\,r$ divided by the total number of tuples. In addition to array H, we maintain the total number of tuples, called the histogram *size*. $LI(n)$ denotes the histogram used as the local index of node n and $S(LI(n))$ its size.

Besides its local index, each node n maintains one routing index $RI(n, e)$ for each of its links e, that summarizes the content of all nodes that are reachable from n using link e at a distance at most R, called *radius*. Routing indexes (e.g., [6]) are used to route queries towards nodes that are expected to store many results for a query. The histogram-based routing index is computed as follows:

Definition 3 (histograms as routing index). *The histogram-based routing index $RI(n, e)$ of radius R of the link e of node n is defined as follows: for $0 \le i \le b - 1$,*

$$RI_i(n, e) = \frac{\Sigma_{p \in P} LI_i(p) S(LI(p))}{\Sigma_{p \in P} S(LI(p))}, \quad S(RI(n, e)) = \Sigma_{p \in P} S(LI(p)),$$

where P is the set of nodes p within distance R of n reachable through link e.

An example is shown in Fig. 2. The set of nodes within distance R of n is called the *horizon* of radius R of n. For a given query q, the local histogram $LI(n)$ of a node n provides an estimation of the number of results (matching tuples) of node n, while the routing index $RI(n, e)$ provides an estimation of the number of results that can be found when the query is routed through link e up to a distance R. We shall use the notation $H(n)$ to denote a histogram (used either as a local $LI(n)$ or as a routing index $RI(n, e)$).

We consider a workload of range queries over attribute x: W ={(q_{ij}, f_{ij}), where q_{ij} = {t: $i \le t.x < j$} and f_{ij} its frequency}. The *range* of the query is $j - i$. For a query q, let $hresults(H(n), q)$ be the number of matching tuples estimated using the histogram $H(n)$ of node n. Using histograms to estimate the number of results is straightforward [16].

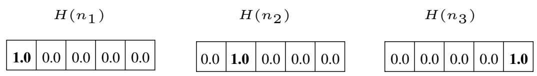

Figure 3: Intuitively, the distance between $H(n_1)$ and $H(n_2)$ should be smaller than the distance between $H(n_1)$ and $H(n_3)$.

3.2 Histogram Distance Metrics

We consider two-well known distance metrics (namely, the L_1 and the edit distance) and extend them to take into account the workload. For simplicity of presentation, in the following we consider equal-sized histograms. The L_1-distance of two histograms $H(n_1)$ and $H(n_2)$ is defined as:

DEFINITION 4 (L_1 HISTOGRAM DISTANCE). *Let $H(n_1)$ and $H(n_2)$ be two histograms with b buckets, their L_1 distance, $d_{L_1}(H(n_1), H(n_2))$ is defined as: $d_{L_1}(H(n_1), H(n_2))$ $= \Sigma_{i=0}^{b-1} |H_i(n_1) - H_i(n_2)|$.*

It is easy to see that the L_1 distance satisfies the workload-awareness property (i.e., it is representative of the number of results) only for very specific types of workloads, in particular, for the workload $W = \{(q_{ij}, f_{ij})$, where $i = mr$, $j = (m+1)r - 1$ for $0 \leq m \leq b-1$ and $f_{ij} = 1/b\}$. One reason is that, although the histograms we study are ordinal (i.e, there is an ordering among their buckets), the L_1 distance does not take into account this ordering. For example, for the three histograms of Fig. 3, the distance between histograms $H(n_1)$ and $H(n_2)$ that have all their values at adjacent buckets (buckets 0 and 1 respectively) should be smaller than the distance between histograms $H(n_1)$ and $H(n_3)$ that have their values at buckets further apart (buckets 0 and 4 respectively). This is because, the difference between the number of results provided by node n_1 and the number of results provided by node n_2 is smaller for a larger number of different range queries than for nodes n_1 and n_3. However, the three histograms have the same pair-wise L_1 distance.

In general, a histogram distance that takes into account the position of buckets is said to satisfy the *shuffling dependence* property. As shown, L_1 does not satisfy this property. We consider an edit distance based similarity metric for which the shuffling dependence property holds. The edit distance of two histograms $H(n_1)$ and $H(n_2)$ is the total number of all necessary minimum movements for transforming $H(n_1)$ to $H(n_2)$ by moving elements to the left or right. It has been shown that this is captured by the following definition [4]:

DEFINITION 5 (EDIT HISTOGRAM DISTANCE). *Let $H(n_1)$ and $H(n_2)$ be two histograms with b buckets, their edit distance, $d_e(H(n_1), H(n_2))$ is defined as: $d_e(H(n_1), H(n_2)) = \Sigma_{i=0}^{b-1} |\Sigma_{j=0}^{i}(H_j(n_1) - H_j(n_2))|$.*

Let us define as:

$$pref(l) = \begin{cases} \Sigma_{i=0}^{l} H_i(n_1) - \Sigma_{i=0}^{l} H_i(n_2), & \text{if } 0 < l \leq b-1 \\ 0, & \text{otherwise.} \end{cases}$$

Then, the edit distance can be written as:

$$d_e(H(n_1), H(n_2)) = \Sigma_{l=0}^{b-1} |pref(l)|. \quad (2)$$

It can be shown that the edit distance satisfies the workload-awareness property for the special case of equi-probable *prefix-range* queries, that is for queries with one sided ranges: $W = \{(q_{0m}, f_{0m})$, where $m = jr - 1$, for $0 < j \leq b$ and $f_{0m} = 1/b\}$. The edit distance is shuffling dependent and workload-aware for some queries whose range spans more than one bucket. However, it does not take into account the frequency of queries, and it is not workload-aware for all possible workloads of range queries.

Recall that the workload-awareness property requires that the distance between the histograms of two nodes be descriptive of the difference in the number of results the nodes provide for a given workload. In the ideal case, we want the distance between two histograms $H(n_1)$ and $H(n_2)$ of nodes n_1 and n_2 to correspond to the difference in the number of results provided by n_1 and n_2 for a given workload. For simplicity, consider a query q_{ij}, where i and j are bucket boundaries. It can be easily shown [16] that the estimated difference in the results, $hdiffer$, of two nodes n_1 and n_2 is:

$$hdiffer(n_1, n_2, q_{ij}) = |pref(j) - pref(i-1)| \quad (3)$$

Based on Eq. (3), we define a workload-aware variation of the edit-based distance metric.

DEFINITION 6. [WORKLOAD-AWARE EDIT HISTOGRAM MEASURE] *Let $H(n_1)$ and $H(n_2)$ be two histograms with b buckets, their workload-aware edit measure, $wd_e(H(n_1), H(n_2))$ is defined as:*

$$wd_e(H(n_1), H(n_2)) = \Sigma_{i=0}^{b-1} \Sigma_{j=i}^{b-1} w_{ij} |pref(j) - pref(i-1)|$$

where $0 \leq w_{ij} \leq 1$ and $\Sigma_{i=0}^{b-1} \Sigma_{j=i}^{b-1} w_{ij} = 1$.

It can be shown [16] that wd_e: (i) is a metric and (ii) satisfies Property 1 for a workload $W = \{(q_{ij}, f_{ij})\}$, when i and j are bucket boundaries, if we set w_{ij} to be proportional to f_{ij}. Our experimental results indicate that it also works well for more general workloads, i.e., for workloads where i and j are not necessarily bucket boundaries.

3.3 Multi-Attribute Histograms

So far, we have considered queries that involve a single attribute. Histogram-based local and routing indexes can be used also in the case of queries that involve more than one attribute. Let us assume m attributes x_i, $1 \leq i \leq m$ each having (without loss of generality) a numerical domain $\mathcal{D}_i = \{0, \mathcal{M}_i - 1\}$. There is a number of different approaches to exploiting histograms in this case. One approach is to select one of the attributes and cluster on this. The attribute selected may be the one present in most queries or the one with the largest skew among nodes. Another straightforward extension is to select c ($1 < c \leq m$) attributes and build one histogram per each. Then, we can define the distance between two nodes as the weighted sum of the c single-attribute *workload-aware* distances. The weight for each of the distances should be based on the popularity of the respective attribute.

Such approaches, however, ignore any value dependencies that may exist among the attributes. To deal with this issue, we may also consider a multi-dimensional histogram built on all or a subset c of the m attributes. Again, we consider equi-width histograms. Each $\mathcal{D}_i$ is divided into b_i ranges of equal width r_i. The buckets are now hyper-rectangles. The value of bucket $H_{i_1 i_2 \ldots i_c}$ is the percentage of items having values $i_1\, r_1 \leq t.x_1 < (i_1+1)\, r_1$ and $i_2\, r_2 \leq t.x_2 < (i_2+1)\, r_2$ and $\ldots$ and $i_c\, r_c \leq t.x_c < (i_c+1)\, r_c$.

The distances defined in the previous section can be generalized for multidimensional histograms. For example, let us define $pref(l_1, l_2)$ for the case of two-dimensional histograms $H_{ij}(n_1)$ and $H_{ij}(n_2)$ for two attributes x_1 and x_2. $pref(l_1, l_2) = \Sigma_{i=0}^{l_1}\Sigma_{j=0}^{l_2}H_{ij}(n_1)$ - $\Sigma_{i=0}^{l_1}\Sigma_{j=0}^{l_2}H_{ij}(n_2)$ for $0 < l_1 \leq b_1 - 1$ and $0 < l_2 \leq b_2 - 1$ and $pref(l_1, l_2) = 0$, otherwise. Consider the range query $q_{i_1 j_1 i_2 j_2} = \{r\text{: } i_1 \leq t.x_1 < i_2$ and $j_1 \leq t.x_2 < j_2\}$. It holds:

$hdiffer(n_1, n_2, q_{i_1 j_1 i_2 j_2}) =$
$|pref(i_2, j_2) - pref(i_2, j_1 - 1) - pref(i_1 - 1, j_2) + pref(i_1 - 1, j_1 - 1)|$.

Based on the equation above, similar to Def. 6, we get the workload-aware variation of the edit distance for two-dimensional histograms.

4. BUILDING THE OVERLAY

We describe first how histogram-based routing indexes are used to route a query and then how workload-aware small-world overlays are constructed. We distinguish between two types of links: *short-range* or short links that connect nodes with similar histograms and *long-range* or long links that connect nodes with non-similar histograms.

4.1 Query Routing

A query q may be posed at any node n. Our goal is to route the query q through a set of nodes that gives a large number of results for q, that is, we want to maximize $PeerRecall$. To achieve this, we use the following simple heuristic: each node n that receives a query q propagates the query through the link e whose routing index gives the most matches ($hresults(RI(n, e), q) \geq hresults(RI(n, l), q)$, $\forall$ link $l \neq e$). By following this link, the query is propagated towards the nodes that are estimated to provide the most results and thus $PeerRecall$ is increased. The routing of a query stops either when a predefined number of nodes is visited ($MaxVisited$) or when a satisfactory number of results is located. Initially, if there are no matching nodes within the horizon of a node ($hresults(RI(n, e),\ q) = 0\ \forall$ link e of n), we follow the long-range link of node n (even if it does not match the query). The idea is that we want to move to another region of the network, since the current region (bounded by the horizon) has no matching nodes.

4.2 Overlay Construction

Each new node n that enters the system tries to locate similar nodes to connect with using its local index $LI(n)$ as a query. In particular, each new node n posses to some known node in the system a join message that contains its local index $LI(n)$ as a query. With the join message, a list L (initially empty) is also maintained with all nodes visited during the routing of the message. Each node p that receives this message propagates it through its link e whose routing index $RI(p, e)$ is the most similar with $LI(n)$ ($d(RI(p, e), LI(n))$ $\leq d((RI(p, l), LI(n))\ \forall$ link $l \neq e$). The message is propagated until up to $JMaxVisited$ nodes are visited. Then, node n creates short links with the SL nodes in the list L of visited nodes whose local indexes have the smallest distance with it. With probability P_l, node n also creates a long link with another node from the list different from the SL nodes previously selected. Short links are inserted so that nodes with relevant data are located nearby in the overlay and a large clustering coefficient is attained. Long links are used for keeping the network diameter small. The motivation is that we want to be easy to find both all relevant results once in the right group, and the relevant group once in another group, thus increasing $PeerRecall$.

When a node n leaves the system, there are two issues involved. Due to space limitation, we provide just an outline of their treatment here. First, the local index of node n is "subtracted" from the routing indexes of all nodes in its horizon. To achieve this, the local index $LI(n)$ is propagated to all nodes within network distance R (i.e, the radius) of n. Second, we must ensure that the network remains connected and that the small-world properties still hold. For keeping the network connected, the idea is to link in a path all the neighbors of the departing node n. Further, for ensuring that the small-world properties hold, node n selects some of the nodes in its group and moves its long links (if any) to them. In addition, it replaces its short links to neighboring nodes by connecting each such neighboring node to some other node in its group.

4.3 Workload Estimation

An important issue is the estimation of the query workload. This issue is beyond the scope of the paper, however, we outline some ideas of attaining such estimations. Some or all nodes in the system may keep statistics of the queries that arrive to them, that is, of their local query workload. For the estimation of this workload, each node may keep a *random sample*. The idea is for each set S of M queries arriving at a node, to sample m ($<< M$) queries randomly and uniformly and replace the previous sample of this node. Another approach to obtaining the sample of the local query workload is the *reservoir sampling algorithm* [23]. For each set of M queries arriving at a node, a random sample of s queries is computed as follows. Each node inserts the first s queries from the set S into a "reservoir". Then a random number of queries from S are skipped and the next query replaces a randomly selected query in the reservoir. Another random number of queries is then skipped, and so forth, until the last query from the set S arrives at the node. The same procedure is repeated with the next set S of M queries arriving at the node. To obtain an estimation of the total query workload, some form of gossiping or epidemic propagation [8] may also be used. Each node must propagate its local workload to a selected subset of its neighbors.

5. PERFORMANCE EVALUATION

In this section, we evaluate the performance of our approach experimentally. We first consider how the distance metrics behave for different types of queries and data distributions. Then, we use the histograms and the different distance metrics to construct and query a workload-aware peer-to-peer system.

5.1 Histogram Similarity

We run a set of experiments to evaluate the histogram distance metrics with regards to the workload-awareness property (Property 1). We use 100 histograms (nodes) $H[m]$, $0 \leq m < 100$, with 100 buckets each and $x \in [0, 999]$. There are 100000 tuples per histogram (node). We consider a data distribution where for histogram $H[m]$, $0 \leq m < 100$, a fraction of the tuples, denoted as DC (*Data Concentration*), falls in bucket m and the remaining ones are distributed uniformly among the rest of its buckets. We considered other distributions as well, where the majority of tuples are dis-

Table 1: Input parameters

Parameter	Default Value	Range
Number of nodes ($\|N\|$)	500	500-1500
Radius of the horizon	2	1-3
Number of short links (SL)	2	1-2
Probability of long link (P_l)	0.4	0.2-1
Nodes visited during join ($JMaxVisited$)	$\log(\|N\|)$	
Perc of nodes visited during routing ($MaxVisited$)	5	
Histogram-Related Parameters		
Number of buckets (b)	100	
Domain of x	[0, 999]	
Tuples per node	100000	
Range of queries	0, 10	0-50
Data Concentration (DC)	0.7	0.5-0.9

tributed in more than one bucket; the results attained are qualitative the same.

We compute the distance of each histogram $H[m]$, $1 \leq m < 100$ (i.e. the histogram with most tuples in bucket m), with $H[0]$ (the histogram with most tuples in bucket 0) using the three distance metrics. We consider the difference between the number of results estimated by each $H[m]$ and the number of results estimated by $H[0]$, that is: $hdiffer(H[m], H[0], q_{ij})$ with respect to their distance $d(H[m], H[0])$. The desired behavior (as expressed by Property 1) is for the difference in the number of results estimated by the two histograms to be proportional to their distance: the smaller the distance between the two histograms, the smaller the difference in the result sizes.

We consider the average performance of the three distance metrics for a mixed query workload of queries with range from 0 to 50 (Fig. 4). The weights w_{ij} of the workload-aware edit distance are set equal to the query frequencies: $w_{ij} = f_{ij}$, where f_{ij} is the frequency of the queries starting at i and ending at j and thus having range $k = j - i$. The starting bucket of each query is chosen uniformly at random. We also consider the case where the starting point i does not correspond to a bucket boundary but is chosen uniformly at random from the domain of the attribute ($wedit(general)$).

As expected, L_1 does not perform well. In particular, because of the nature of the data distribution, all histograms have the same L_1 distance with $H[0]$, since L_1 considers only individual buckets. However, their result sizes when compared with $H[0]$ are different (for all queries but for those with range equal to 0). The edit distance performs better than L_1, since it takes into account the order of buckets; it is shuffling-dependent. In particular, as the distance between the histograms increases, their respective difference in the results also increases. However, at some point, the difference in the results becomes constant while the histogram distances do not. This is because the edit distance between two histograms takes into account the ordering of all buckets, while a query with range k involves only $k + 1$ buckets, and thus it does not depend on the difference that the two histograms may have in the rest of their buckets. For example, for a query with range 0, the difference in the results between any $H[m]$ and $H[0]$ is the same, since the query involves only a single bucket. However, their edit distance is different, since it considers all their buckets. Consequently, the edit distance works well only for queries with

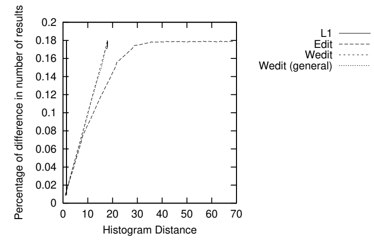

Figure 4: Comparison of histogram distances.

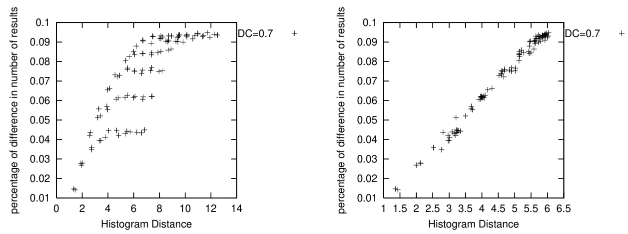

Figure 5: (left) Edit and (right) workload-aware edit distance for multi-attribute histograms

large ranges. For the workload-aware edit distance, the difference in the results increases proportional to the histogram distances. The workload-aware edit distance satisfies the workload-awareness property even when i is not a bucket boundary. We also consider the sensitivity of the results with regards to the choice of DC. The results are similar.

We then conducted a number of experiments using multi-attribute queries. In this case, each node stores a relation that includes tuples of two integer attributes $x_1, x_2 \in [0, 99]$. For ease of presentation, we consider 10 buckets per dimension, resulting in histograms with 100 two-dimensional buckets. Again, we use a data distribution where 70% of the tuples of each node belong to one bucket, and the rest are uniformly distributed among the rest of the buckets. The tuples in each bucket also follow the uniform distribution. Figure 5 demonstrates that the performance of (left) the edit and (right) the workload-aware edit distance is similar to that of single-attribute histograms.

5.2 Building the Overlay

In this set of experiments, we use histograms to construct a small-world overlay network. We consider a p2p network where each node stores tuples with an integer attribute $x \in [0, 999]$. The tuples at each node are summarized using a histogram with 100 buckets. We consider the same data distribution as in the previous experiments with DC set to 0.7. We study the properties of the network created using our workload-aware small-world construction procedure, in particular, we evaluate the clustering quality and the diameter of the network. When joining the network, each new node creates 1 to 2 short links ($SL = 1$ or 2) and one long link with varying probability P_l. The routing of the join message stops when a maximum number ($JMaxVisited$) of nodes is visited. $JMaxVisited$ is set equal to a small fraction of the nodes in the network roughly equal to $log(|N|)$ where $|N|$ is the number of nodes. We compare the small-world p2p networks constructed using the three distance metrics with a randomly constructed p2p network, that is with a p2p system in which each new node connects to the same number of nodes as in the small-world network but these nodes are

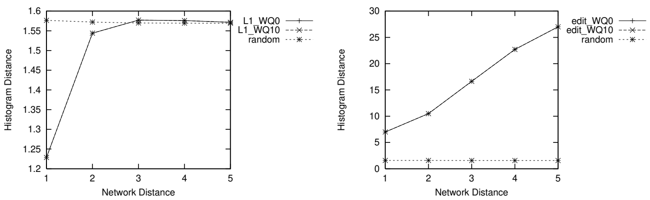

Figure 6: Clustering quality for (left) L_1 and (right) edit distance.

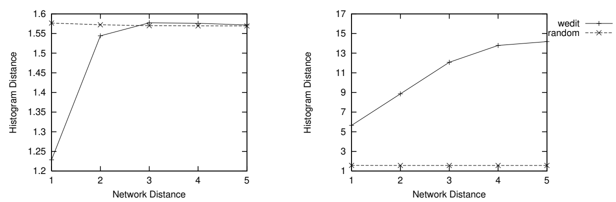

Figure 7: Clustering quality for workload-aware edit distance for (left) QW_0 and (right) QW_{10}.

chosen uniformly at random from the nodes currently in the system. The size of the network varies from 500 to 1500 nodes and the radius of the horizon from 1 to 3. The input parameters are summarized in Table 1.

To demonstrate the workload-aware characteristics of the p2p network, we consider two different types of query workloads: (a) QW_0 with queries with range equal to 0 and (b) QW_{10} with queries with range equal to 10. The L_1 distance is expected to work well with QW_0, since it considers individual buckets, while the edit distance is expected to perform better with QW_{10}. The workload-aware distance should work well in both cases, since its weights are adapted based on the workload. Figures 6 and 7 show the clustering quality of the created overlays. The p2p network created by using the L_1 and the edit distance is independent of the query workload (Fig 6), thus, it is the same for both the QW_0 and the QW_{10} workloads. However, when using the workload-aware edit distance, the network created for QW_0 (Fig. 7(left)) is different than the one created for QW_{10} (Fig. 7(right)).

For all distance metrics, the diameter of the network created is small (of logarithmic order to the number of nodes, i.e. below 10 for $SL = 2$ and 500 nodes), very close to that of the random network. In terms of clustering, in the created network, nodes with similar histograms should be located nearby in the network. To evaluate this, we measure the average histogram distance between nodes at the various network distances in the created p2p networks. For the QW_0 workload, the desirable clustering is the one that clusters together nodes that have the majority of their tuples at the same bucket. There is no need for ordering among these clusters, that is, all clusters are "similar" to each other with respect to the results they return to QW_0. As shown in Fig. 6(left), this is achieved by the L_1 distance. For the QW_{10} workload, the desirable clustering is the one that clusters together nodes that have the majority of their tuples at up to 10 neighboring buckets. There should be some ordering among these clusters, since some clusters are "more similar" than others in terms of the results they return to QW_{10}. As shown in Fig. 6(right), the edit distance orders nodes based on the similarity of all their buckets (not just the 10 adjacent ones). The workload-aware distance behaves as the L_1 for the QW_0 workload (Fig. 7(left)). For the QW_{10} workload, it orders nodes based on their adjacent buckets (Fig. 7(right)). For the random network, the histogram distance between two nodes is the same independently of their network distance, since there is no clustering of similar nodes.

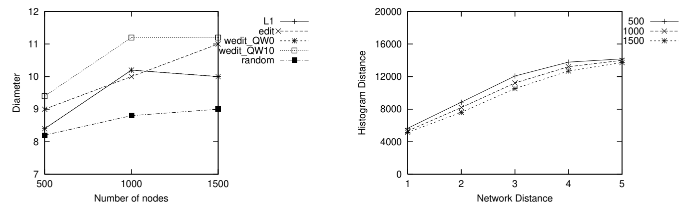

Figure 8: Influence of the number of nodes on the (left) diameter and (right) clustering quality.

We performed various experiments with different parameters for the network. The results are similar for $SL = 1$. However, the clustering created is more weak since there are fewer links among similar nodes than for $SL = 2$. This becomes more evident for the edit and the workload-aware edit distance for QW_{10} where the ordering of the nodes based on their histogram distance stops after some network distance. Thus, for $SL = 1$ due to the limited number of links in a group, similar nodes may be located further apart. We also performed a number of experiments varying the number of nodes in the network from 500 to 1500. workload. As shown in Fig. 8(left) for the QW_{10} query workload, the diameter of the network scales well when increasing the number of nodes and remains of logarithmic order. Figure 8(right) depicts the clustering quality that remains unaffected as the number of nodes increases, which means that examining a small number of nodes when joining the system suffices.

Query Routing. In this set of experiments, we evaluate the performance of query routing using as our performance measure *PeerRecall* (as defined in Def. 1). We compare the constructed clustered network with a randomly constructed p2p system (*random*) and also with a randomly constructed p2p system that uses histograms for query routing only (*random_join*). We consider a network of 500 nodes and examine the influence of the horizon in the query routing performance for both the QW_0 and QW_{10} query workloads. The radius varies from 1 to 3. Query routing stops when a maximum number ($MaxVisited$) of nodes is visited; this is set equal to roughly 5% of the nodes.

Using histograms for both clustering and query routing results in much better performance than using histograms only for routing or not using histograms at all (Fig. 9). *PeerRecall* decreases for radius greater than 2. The reason is that there are more links, and thus, much more nodes are included within the horizon of a particular node. Thus, a very large number of nodes correspond to each routing index. This results in losing more information than using radius 2.

We performed various experiments regarding a number of parameters of the network construction procedure. Since the focus of this paper is on the workload-aware properties of the network, we report only briefly on some of our findings. We varied the probability P_l of creating a long link for each node that joins the system from 0.2 to 1. In general, *PeerRecall* improves as the number of long links

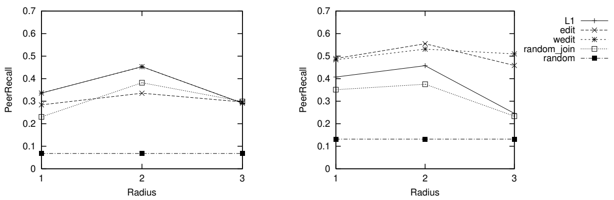

Figure 9: ***PeerRecall*** **when varying the radius with** SL **= 2 for (left)** QW_0 **and (right)** QW_{10}**.**

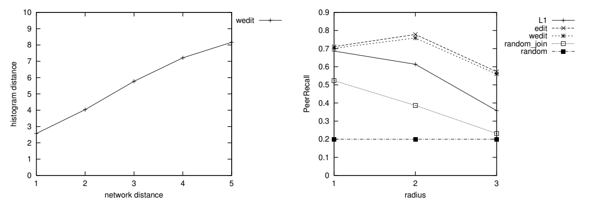

Figure 10: (left) Clustering quality (radius = 2) and (right) ***PeerRecall*****.**

increases, since it is easier to locate the appropriate group. However there is a value of P_l after which there is no significant improvement in performance. This value depends on the network configuration (e.g., number of short links), the histogram distance used, the data distribution and the query workload. In general, a value of P_l below 0.6 seems to work well. We also varied the number of nodes from 500 to 1500, *PeerRecall* remains nearly constant for all histogram distance metrics. Finally, we also evaluated the performance of the network (clustering and *PeerRecall*) as nodes leave the system. Both clustering and *PeerRecall* remain unaffected even when 50% of the nodes leave the network.

Using Multi-Attribute Histograms. We conducted a set of experiments using multi-attribute histograms. Tuples are distributed among the nodes as in the single-attribute case. Our experimental results show a behavior similar with the single-attribute case. Figure 10 depicts the network constructed for a two-dimensional histogram and for a query workload with range 3 for each attribute (whose results occupy 16 buckets). Again, the workload-aware edit distance provides the best results. Figures 10(left) and 10(right) show the clustering quality of the constructed network and its routing performance respectively, for 500 nodes and $SL = 2$.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we propose using histograms for building small-worlds overlays. The creation of the small worlds is workload-aware, since it depends on the query workload. To achieve this, we introduce an appropriate workload-aware histogram distance metrics that incorporate the frequency and range of queries in the definition of distance. Our experimental results show that our procedure is effective: in the constructed overlay, the network distance of two nodes is proportional to the distance of their local indexes and thus nodes that match similar queries are nearby in the network. Furthermore, routing is very efficient, in particular, for a given number of visited nodes, the results returned are 60% more than in a non-clustered overlay. We are currently working on techniques for workload estimation and for adapting the overlay as the system workload changes. We are also looking into using other type of histograms besides equi-width ones as well as using histograms in structured p2p systems.

7. ACKNOWLEDGMENTS

Work supported by EPEAEK II, program Pythagoras.

8. REFERENCES

[1] K. Aberer, P. Cudré-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Punceva, and R. Schmidt, P-Grid: A Self-organizing Structured P2P System. *Sigmod Record*, 32(2), 2003.

[2] M. Bawa, G. S. Manku, and P. Raghavan. SETS: Search Enhanced by Topic Segmentation. In *SIGIR*, 2003.

[3] A. R. Bharambe, M. Agrawal and S. Seshan. Mercury: Supporting Scalable Multi-Attribute Range Queries. In *SIGCOMM*, 2004

[4] S-H Cha and S. N. Sribari. On Measuring the Distance Between Histograms. *Pattern Recognition*, 35:1355–1370, 2002.

[5] E. Cohen, A. Fiat and H. Kaplan. Associative Search in Peer to Peer Networks: Harnessing Latent Semantics. In *INFOCOM*, 2003.

[6] A. Crespo and H. Garcia-Molina. Routing Indices for Peer-to-Peer Systems. In *ICDCS*, 2002.

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

[8] A. J. Demers, D. H. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. E. Sturgis, D. C. Swinehart and D. B. Terry. Epidemic Algorithms for Replicated Database Maintenance. In *PODC*, 1987.

[9] P. Ganesan, B. Yang, and H. Garcia-Molina. One Torus to Rule Them All: Multidimensional Queries in P2P Systems. In *WebDB*, 2004.

[10] R. Morris I. Stoica, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. *IEEE/ACM Trans. on Networking*, 11(1):17–32, 2003.

[11] A. Iamnitchi, M. Ripeanu, and I. T. Foster. Locating Data in (Small-World?) Peer-to-Peer Scientific Collaborations. In *IPTPS*, 2002.

[12] Y. Ioannidis. The History of Histograms. In *VLDB*, 2003.

[13] A. Loser, F. Naumann, W. Siberski, W. Nejdl, and U. Thaden. Semantic Overlay Clusters within Super-Peer Networks. In *DBISP2P*, 2003.

[14] S. Manku, M. Bawa, and P. Raghavan. Symphony: Distributed Hashing in a Small World. In *USENIX Symposium on Internet Technologies and Systems*, 2003.

[15] Y. Petrakis, G. Koloniari, and E. Pitoura. On Using Histograms as Routing Indexes in Peer-to-Peer Systems. In *DBISP2P*, 2004.

[16] Y. Petrakis, G. Koloniari, T. Tsotsos, and E. Pitoura. On Constructing Workload-Aware Small-Worlds in Peer-to-Peer Systems (extended version of this paper). Technical Report TR2004-14, Univ. of Ioannina, Dept. of Computer Science, Oct 2004.

[17] V. Poosala, Y. Ioannidis, P. Haas, and E. Shekita. Improved Histograms for Selectivity Estimation of Range Queries. In *SIGMOD, 1996*

[18] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A Scalable Content-Addressable Network. In *SIGCOMM*, 2001.

[19] O. D. Sahin, A. Gupta, D. Agrawal, and A. El Abbadi. A Peer-to-Peer Framework for Caching Range Queries. In *ICDE*, 2004.

[20] C. Schmidt and M. Parashar. Flexible Information Discovery in Decentralized Distributed Systems. In *HPDC*, 2003.

[21] C. Tang, Z. Xu, and S. Dwarkadas. Peer-to-Peer Information Retrieval Using Self-Organizing Semantic Overlay Networks. In *SIGCOMM*, 2003.

[22] P. Triantafillou, C. Xiruhaki, M. Koubarakis, and N. Ntarmos. Towards High Performance Peer-to-Peer Content and Resource Sharing Systems. In *CIDR*, 2003.

[23] J. S. Vitter. Random Sampling with a Reservoir. ACM Trans. Math Software, 11:37-57,1985.

[24] D. J. Watts and S. H. Strogatz. Collective Dynamics of Small-World Networks. *Nature*, 393:440–442, 1998.