Workload-Aware Clustering of XML Peers

Georgia Koloniari
Computer Science Department
University of Ioannina, Greece
kgeorgia@cs.uoi.gr

Evaggelia Pitoura
Computer Science Department
University of Ioannina, Greece
pitoura@cs.uoi.gr

Abstract

Peer-to-peer (p2p) systems are attracting increasing attention as an efficient means of sharing data among large, diverse and dynamic sets of users. Clustering in p2p systems aims at improving query processing performance by reducing communication costs, through placing similar data at neighboring peers. In this paper, we present a distributed procedure for p2p clustering in a system where peers share collections of XML documents. Peer clustering is based on the use of rules that associate the existence of path patterns in a peer's data to its likelihood of belonging to a given cluster. The patterns that form the rules are selected both from the content of the peers and the query workload of the system according to their popularity. Furthermore, we describe how these rules adapt to reflect changes in the content of the peers and the local query workload of each cluster.

1. Introduction

Peer-to-peer (p2p) systems have emerged as an intelligent way of exploiting the vast computing power and storage capacity that remains unused and scattered around multiple nodes in the Internet. Through the cooperation of a multitude of autonomous nodes, p2p systems provide the means for efficient data and services sharing to a large set of diverse and dynamic users. The peers form logical overlay networks by establishing links to some other peers they know or discover and use these logical links to forward their queries and requests so as to retrieve the data of their interest.

Meanwhile, XML [24] that has evolved as the new standard for the representation and exchange of semistructured data on the Internet seems a natural choice for describing and querying the resources and data in p2p systems. The flexibility of XML in describing heterogeneous data makes it suitable for distributed applications where the data are either native XML documents or XML descriptions of data or services that are represented in different formats in the underlying sources (i.e. in relational databases).

Clustering has been recently proposed for improving the performance in p2p systems, by reducing communication costs through placing similar data at neighboring peers. We can distinguish between (i) clustering similar data items or indexes of similar data items so that similar data or indexes of similar data respectively are placed in neighboring peers and (ii) clustering peers with similar data items, so that their distance in the overlay network is small. By grouping similar peers, a query that reaches a peer in the cluster finds all other peers with relevant data nearby. Clustering the peers affects the topology of the p2p overlay. Usually, query routing proceeds in two steps: first the appropriate cluster is identified and then the query is routed inside the cluster.

In this paper, we present a distributed workload-aware procedure for clustering peers in a p2p system that uses XML as the underlying data model. In particular, the system follows a hybrid p2p architecture where each superpeer acts as the representative of a cluster and is responsible for maintaining the cluster description. The cluster description is defined as a set of path expressions that best describe both the content of the peers that belong in each cluster and the local query workload of the cluster, i.e., the queries from the system query workload that are satisfied by the peers in the cluster. The representative path expressions that are included in each description are selected based on their popularity, since the most popular expressions are those that best describe the contents of each cluster.

Peer clustering is based on rules that associate the presence of each path expression of a cluster description in the documents of the peers to the likelihood of the peer belonging to this cluster, an idea presented in XRules [25], a centralized algorithm for clustering XML data that uses discriminatory structural patterns to determine the cluster each document belongs to. Furthermore, clustering is adaptive in the sense that each cluster description changes over time as new peers join the system and share new data and as the query workload of the system changes.

The key contributions of our clustering procedure are that:

- no global knowledge of the distribution the documents of the participating peers follow is required,
- the clusters are created according to both the content of the peers and the query workload of the system and
- the cluster descriptions change over time to reflect changes in the peers contents and the query workload.

The rest of the paper is organized as follows: in Section 2, we present related work. In Section 3, we outline the data and query language model that is supported in the system along with the overall system architecture. In section 4, we describe our rule-based clustering algorithm and describe how the cluster description is derived only from the content of the peers. We also present the peer join and the query routing procedures. In Section 5, we present how the cluster description is adapted when we have knowledge about the workload of the system. In Section 6, we present our experimental results and we conclude in Section 7, with a summary and directions for future work.

2. Related Work

Our system is based on creating clusters of nodes that have similar data or satisfy similar sets of queries. Such p2p systems can be neither classified into *structured* p2p systems that are usually based on distributed hash tables ([20], [17]) and confine peers into rigid topologies, nor into *unstructured* p2p systems ([13], [6]) which are organized in an ad hoc manner.

With regards to clustered p2p systems, in most current research, the number or the description of the clusters is fixed and global knowledge of this information is required. Furthermore, the schema that the data of the peers follow or a taxonomy in which they belong needs to be predefined so as to determine the created clusters. With Semantic Overlay Networks (SONs), peers with semantically similar content are logically linked to form overlay networks based on a classification hierarchy of their data, which is defined a priori [4]. Queries are processed by identifying which SONs are better suited to answer them. In [23], clusters of peers are again formed based on the semantic categories of their data. Sophisticated procedures are proposed for both inter-cluster and intra-cluster load balancing. Similarly in [2], peers are partitioned into topic segments based on their data. A fixed set of C clusters is assumed, each one corresponding to a topic segment. Knowledge of the C centroids is global.

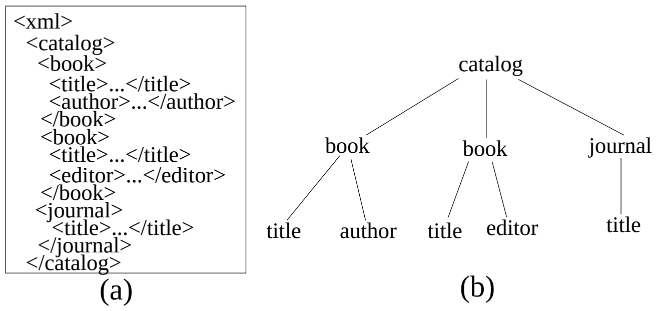
In structured p2p systems, clustering the data or index can be achieved by using as input to the hash function not just the name of the document but a semantic vector describing its content and structure. If the hash function is order-preserving, similar documents are stored at the same or neighboring peers. Order preserving hash functions are those hash functions that for similar inputs they produce outputs close in the identifier space. When schema information is available, the virtual address space can be split to sub-spaces each one corresponding to a different part of the global schema ([19], [14]). Then, upon entering the system, each peer (or data item) can be mapped to the sub-space of the virtual space that corresponds to the schema of the peer, thus creating clusters of peers that follow the same schema.

Schema information has also been exploited in hybrid p2p systems where the superpeers collect their peers' schemas and peers form clusters under the superpeers on a schema similarity basis. In SQPeer [7], peers are grouped based on their RDF-schema similarity. The peers that hold RDF descriptions conforming to the same RDF schema are clustered together. In XPeer [18], peers are logically organized into clusters that are also based on schema-similarity, whenever this is possible. Superpeers are organized to form a tree, where each peer hosts schema information about its children. In rule-based clustering [11], peers are registered and grouped in clusters based on cluster specific rules that describe the properties that each peer in the cluster should possess. These rules are provided by a cluster's administrator.

Our approach, does not require any knowledge of the schemas the data belong to, nor the use of predefined categories. It organizes peers into clusters based solely on information dynamically extracted from their content and the query workload of the system.

In [8], a form of clustering is applied to an unstructured p2p system of XML peers. The peers are organized into hierarchies according to their content similarity. Content similarity is derived from the similarity of their routing indexes. Similarity takes into account both the structure and the content of documents. Upon entering the system, each peer sends its index to the roots of the hierarchies that compare it with their own indexes. The peer attaches to the most similar hierarchy, so that peers with similar content are organized into the same hierarchy. This approach does not require any knowledge of the schema of the data. However, it does not take into account the frequency of the data items into the content of the peers and it also does not exploit the query workload of the system.

Related research on clustering XML documents in centralized settings, which relies mostly on structural information, cannot be directly applied in a dynamic p2p system where the documents are distributed among multiple nodes. For the classification of schema-less data, the authors of [21] combine text terms, structural information in the form of twigs and paths and also ontological knowledge (WordNet [5], [12]) to construct more expressive feature spaces that are then used for the classification. S-GRACE [10] is a



```
<xml>
  <catalog>
    <book>
      <title>...</title>
      <author>...</author>
    </book>
    <book>
      <title>...</title>
      <editor>...</editor>
    </book>
    <journal>
      <title>...</title>
    </journal>
  </catalog>
```

Figure 1. Example of (a) an XML document and (b) its corresponding XML tree

hierarchical algorithm for clustering XML documents with a distance metric based on the notion of a structure graph, which is a minimal summary of edge containment in the data. In contrast with the previous methods that can be applied to schema-less data, XClust [9] addresses clustering when schema information in the form of DTDs is available. XClust clusters DTDs based on the semantics, immediate descendants and leaf-context similarity of DTD elements. XRules [25] assigns the documents to categories through a rule based classification approach that relates the presence of a particular structural pattern in an XML document to its likelihood of belonging to a particular category. We adopted XRules by dynamically extracting the patterns that are used as the rules, and instead of associating each set of rules to a predefined category we associate it with a cluster.

3. Clustered Overlay Network

A clustered p2p system, is a p2p system in which the peers are organized into groups that form the logical overlay network.

3.1. Data Model and Query Language

We consider a system in which each peer stores a set of XML documents. We model an XML document as a node-labeled tree $T(V, E)$ (Fig. 1). Each node $e_i \in V$ corresponds to an XML element with a label assigned from some some string literals alphabet that captures element's semantics. Edges $(e_i, e_j) \in E$ are used to capture the containment of element e_j under e_i. Although leaf elements in T typically contain values, in our work as a first step we ignore values and mainly focus on the label structure of an XML tree. The system supports queries that use simple XPath expressions involving only the child and descendant-or-self axes (i.e."/" and "//" operators).

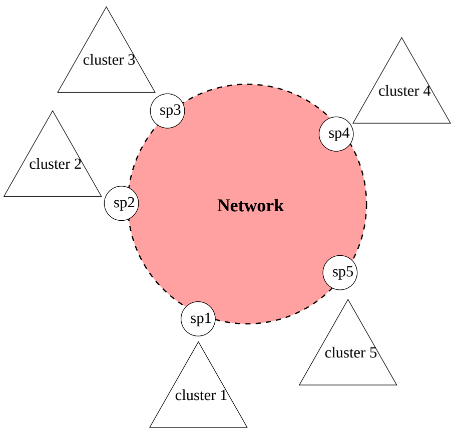


Figure 2. System Architecture

3.2 System Architecture

Our system is based on a hybrid p2p architecture with a number of peers having increased responsibilities, called *superpeers*. The superpeers are usually peers that have increased capabilities (storage, processing, etc) and good stability properties. Each superpeer in the system is responsible for the management of a single cluster and its interaction with the other clusters in the system and acts as that cluster's *representative*. Each cluster representative maintains the cluster description and knows all the peers that belong in its cluster.

The representatives are responsible for routing the queries that concern their clusters to the corresponding peers and also route the queries of the peers of their cluster to the other clusters that may contain relevant results. We assume that the cluster representatives are fully connected with each other forming the backbone of the overlay network, although other topologies are also feasible. Simple peers are organized into clusters by following a star topology, with each simple peer connected to a cluster representative (Fig. 2). Furthermore, the representatives stores the cluster description that is used for query routing and peer join. Apart from its own cluster description, each superpeer also maintains the cluster description of all the other clusters in the system.

4. Rule-based Clustering

The clustering algorithm we apply on the peers is based on the idea presented in the XRules [25] centralized clustering procedure for XML documents. XRules assigns each document to a category from a set of predefined categories, through a rule based classification approach that relates the presence of a particular structural pattern in an XML document to its likelihood of belonging to a particular category.

XRules associates each category with a set of structural patterns and examines the presence of these structural patterns in each document to decide to which category each document belongs.

Based on the same basic idea, we define the cluster descriptions as sets of simple path expressions. These path expressions are chosen so as to be descriptive of the content of the peers and the query workload of each cluster. The goal of the description is to improve the performance of query routing, by efficiently determining the appropriate clusters each query needs to be forwarded to.

In particular, we want to include into the description of each cluster the path expressions that are most descriptive of its peers contents, thus they exist most frequently in the documents of their peers. By including these path expressions into the description, queries can efficiently locate the clusters that contain the larger number of results and retrieve them.

Similarly, we also want to include in the cluster description the most popular queries from the system's query workload that concern the given cluster. By including the most popular queries that are satisfied by the given cluster, we include a description of the data that are more often requested from this cluster. We expect that if similar queries are posed in the future they will be more efficiently routed to this cluster.

To limit the number of path expressions that are included in a cluster's description we can use a predefined maximum number N of path expressions and include in the description the N path expressions with the greater frequency.

Definition 1 ***(Cluster Description)*** *The description of a cluster i, CD_i is a set of N simple path expressions p_1, $p_2, \ldots, p_N$, extracted from the contents of its peers and the query workload that concerns it, sorted by their frequency.*

The path expressions that are included in each cluster description are used as the rules in the clustering procedure. The path expressions of each CD_i are matched against the documents of a peer and the probability of a peer belonging to a given cluster is determined by the existence of the path expressions of the CD_i in the peer's documents. In particular, the similarity between a peer and a cluster is evaluated by the procedure CalculateScore (Alg. 1). For every path p_i in a cluster description CD we check whether this path exists in the set of documents of the peer (D). If it does exist we increase the similarity score by adding the frequency of the appearance of p_i in D. The final similarity score is evaluated by dividing the total score that is calculated after examining each path expression of the CD with the number of total path expressions that belong to D.

To calculate the similarity scores we need to know all the path expressions that exist in the documents of a peer n, along with their frequencies (number of appearances).

Algorithm 1 CalculateScore(CD, N, D)

```
1: $score := 0$
2: for $i = 1$ to $N$ do
3:     if $p_i \in D$ then
4:         $score := score + count(p_i)$
5:     end if
6: end for
7: $score := score/|D|$
8: return $score$
```

Path Label	Frequency
catalog	1
book	2
title	3
journal	1
editor	1
author	1
catalog/book	2
catalog/journal	1
book/title	2
book/author	1
book/editor	1
journal/title	1
catalog/book/title	2
catalog/book/author	1
catalog/book/editor	1
catalog/journal/title	1

Table 1. Path-count table of the XML document of Fig.1

Instead of examining all the documents and record the frequency of their paths each time we want to evaluate a similarity score we can rely on an auxiliary data structure that provides us with the required information. We use a simple structure called the *path count table*.

Definition 2 ***(Path-Count Table)*** *Given an XML document D_j, a path-count table $T(path, count)$ such that for each path $path_i$ in D_j, there is a tuple t_i in T with $t_i : path = path_i$ and $t_i : count = count_i$, where $count_i$ is the number of occurrences (also referred to as frequency) of $path_i$ in D_j.*

Table 1 is the path-count table that is constructed from the document of Fig. 1.

Although the path-count table requires a lot of space and it cannot be searched very efficiently, we use it in our approach so as to demonstrate its feasibility. Other data structures such as [1], [15], [16] that provide means for summarizing XML documents more efficiently and also offer selectivity estimations can be used instead of the path-count table without affecting the rule-based clustering procedure.

4.1. Content-based Cluster Description

When no knowledge of the query workload of the system is available, for example in the bootstrapping of the system, the cluster descriptions have to rely only on the content of the peers that belong to each cluster.

Each cluster representative is responsible for constructing its cluster description. In particular, each peer p in the system creates a path-count table that includes all the paths that appear in its documents, called *Peer Table*. Each cluster representative gathers all the peer tables of its cluster and merges them into one so as to create the *cluster table*. When new peers join the cluster their peer table is merged into the cluster table (Alg. 2). Similarly, when a peer leaves the system, its local table is subtracted from the corresponding cluster table.

Algorithm 2 MergePathTable(ClusterTable, PeerTable)

```
1: while $PeerTable.entry != NULL$ do
2:     $flag = 0$
3:     for all $ClusterTable.entry$ do
4:         if $ClusterTable.entry(path) == PeerTable.entry(path)$ then
5:             $ClusterTable.entry(count) = ClusterTable.entry(count) + PeerTable.entry(count)$
6:             Add peer $n$ to $ClusterTable.entry_i(contributing)$
7:             $flag = 1$
8:         end if
9:     end for
10:    if $flag == 0$ then
11:        Add entry $PeerTable.entry$ to $ClusterTable$
12:        Add peer $n$ to $ClusterTable.entry_i(contributing)$
13:    end if
14: end while
15: return
```

Apart from the path expression and its frequency each entry in a cluster table also contains a list of peers, called $contributing$, which includes the peers that have the given path expression in the entry in their documents.

The cluster table consists of a summary of the paths that appear in all the documents of the peers in the cluster. Thus, it can be used to extract the most popular path expressions from the contents of the cluster that will form the cluster description. Thus, it can be used for the extraction of the path expressions.

We have to note here that this approach does not only count how many different documents have the same full path expression appearing in their data but also it counts the different times the path expression is met in the same document. By selecting the paths with the highest frequency we choose the path expressions that are more frequently met in the peers contents and thus, are more descriptive of the cluster's content.

Algorithm 3 CreateCD((N, ClusterTable)

```
1: sort $ClusterTable$ according to $count$
2: for $i = 1$ to $N$ do
3:     $CD_i < -ClusterTable.entry_i$
4: end for
5: return $CD$
```

The procedure for building the cluster description $CreateCD$ (Alg. 3) takes as input a cluster table ($ClusterTable$) and the maximum number of expressions allowed in a description (N), and outputs a cluster description with the N path expressions sorted by their frequency in the cluster table.

4.2. Peer Join

When a new peer n wishes to join the network it has to find the appropriate cluster to attach to, i.e. the cluster whose peers have the most similar data to its own. To locate the right cluster, n follows the following procedure:

1. It constructs its peer table and sends a join request to one of the super peers (SP) that form the backbone of the network.

2. When a superpeer SP receives a join request, it replies to n by sending to it all the cluster descriptions in the system.

3. When peer n receives the descriptions it applies the $CalculateScore$ procedure for each of the cluster description to obtain its similarity score with each of the clusters. Peer n determines cluster W as the winner, the cluster it wishes to attach to, the cluster that gave the higher similarity score.

4. Peer n sends a join request to the representative of W along with its peer table.

5. When W receives the join request it merges the peer table of n to the cluster table and adds peer n in its cluster.

4.3. Query Routing

Usually, in clustered p2p systems we can discern two different strategies that govern query routing. The first strategy concerns locating the appropriate clusters that contain data relevant to the query. This is achieved by interactions among the different clusters in the system (*intra-cluster* routing). The second concerns the routing of the query within a single cluster (*inter-cluster* routing). In our system, intra-cluster routing exploit the cluster descriptions, while inter-cluster routing relies upon the cluster table structure.

When a peer issues a query, it directly sends the query to its cluster representative. Upon receiving a query from its own cluster, the representative needs to first check whether its own cluster contain relevant results and then locate the appropriate clusters that are more likely to contain results. In order to do so, it matches the query against the cluster descriptions of all the clusters in the system including its own. The representative then forwards the query only to those cluster representatives for whose clusters it had a match, i.e. the path expression forming the query existed in their descriptions. If no cluster description gives a match, then the query routing proceeds by sequentially selecting a cluster and forwarding the query to it until the application requirements are satisfied, that is, the required number of results are attained.

Depending on the application, the representative can use different strategies to propagate the query to the other clusters. If the peer that issues the query is interested only in the first result then the representative sends the query to its own cluster (if the description gave a match) where the result is more likely to be obtained faster by exploiting locality. If the issuing peer is interested in finding all the available results in the system, the query is forwarded to all the matching clusters. Finally, if the issuing peer is interested only in a number $MaxR$ of results the cluster representative may forward it only to the clusters for which their descriptions returned the highest results (the frequency of the corresponding path expressions appearance was the highest) and thus it is expected that they contain more results for the query.

When a cluster representative receives a query from another cluster representative or when a representative decides that a query issued at its cluster concerns its own cluster, inter-cluster routing is deployed. The representative checks it cluster table and propagates the query to all the peers in its cluster that contributed to the paths that the query matched. This way, only the peers that may contain a relevant result are contacted, minimizing the network traffic and the processing cost for the cluster peers. The peers that receive a query first examine the query against their peer table and if the evaluation returns a non-empty result set, they proceed on evaluating the query against their documents. They finally, return their results to their cluster representative. After receiving the results, the representative records the successful query and sends the results to the peer or superpeer that sent the query to it.

5. Workload-Aware Clustering

By consulting the cluster descriptions the cluster representatives route queries to the appropriate clusters and locate the cluster a new peer belongs to. The cluster description can be enhanced by incorporating the most popular path expressions from the query workload. By including in the description, the most popular queries that are satisfied by a given cluster, we include a description of the data that are more often requested from this cluster. We expect that if similar queries are posed in the future they will be more efficiently routed to this cluster.

5.1. Workload Mining

We assume a query workload consisting of queries in the form of simple path expressions. Each cluster representative records the queries that reach it and are satisfied by the peers that belong to its cluster, thus creating the *local query workload*.

Definition 3 ***(Local Query Workload)*** *The local query workload is defined for each cluster in the system as the set of the queries that are satisfied by the given cluster, i.e. they return a non empty result set when routed inside the given cluster.*

In order to record the local query workload efficiently, each cluster representative constructs a path count table that is used for summarizing the local query workload, called the *workload table*.

Definition 4 ***(Workload Table)*** *The workload table of each cluster is as a path count table that consists of the local query workload of the cluster.*

Each query that reaches a cluster and is satisfied by its peers, is inserted into its workload table. Thus, the workload table provides the cluster representative with the information that is needed in order to enhance the cluster description with query workload information, i.e. the path expressions that form the queries along with their corresponding frequency.

5.2. Workload-Aware Cluster Description

The workload table is periodically used along with the cluster table to produce a new cluster description that better describes a cluster after changes have occurred both on the cluster contents or the workload of the system. This can be done either after a period of time or a predefined number of successful queries that have reached the cluster. The new cluster description contains a percentage of its path expressions K that are extracted from the workload table and a set of $N - K * N$ path expressions extracted from its current cluster table. Procedure $AdaptDescription$ (Alg. 4) describes in detail how the new description is produced.

In particular, the procedure first sorts the entries of both the cluster and the workload table in descending order of frequency. Then the entries of the workload table are examined against the old cluster description. If an entry does exist in the old description then it is automatically added to the new cluster description. Otherwise, it is added in the candidate list. The second step of the algorithm inserts the entries from the candidate list into the new description until the number of entries in the new description is equal to $N * K$. The new cluster description is finally filled with the first $N - K * N$ entries of the cluster table that have not already been inserted into it. The final cluster description is a combination of frequent path expressions from the workload and the cluster table.

Algorithm 4 AdaptDescription(ClusterTable, OldDescription, WorkloadTable, K, N)

```
1:  sort ClusterTable according to $count$
2:  sort WorkloadTable according to $count$
3:  $NewDescription := NULL$, $Candidates := NULL, i = 0$
4:  while $i < N*K$ and WorkloadTable.entry!=NULL do
5:      if WorkloadTable.entry $\in$ OldDescription then
6:          Add WorkloadTable.entry in NewDescription
7:          i:=i+1
8:      else
9:          Add WorkloadTable.entry to Candidates
10:     end if
11: end while
12: while $Candidates.entry! = NULL$ and $i < N * K$ do
13:     Add Candidates.entry to NewDescription
14:     i:=i+1
15: end while
16: while $i < N$ and $ClusterTable.entry! = NULL$ do
17:     if $ClusterTable.entry \notin NewDescription$ then
18:         Add ClusterTable.entry to NewDescription
19:     end if
20: end while
21: return $NewDescription$
```

By periodically using the AdaptDescription procedure, the cluster descriptions change to reflect the changes in the cluster contents which are recorded in the cluster table and the local query workload which are recorded in the workload table. Thus, the rules that determine the cluster in which a new peer joins also change, since they are derived from the cluster description. The new peers that join the system are thus appointed to the cluster that best matches their contents according to the current conditions in the system and not a static snapshot of an older state of the system as is the case in the content-based method.

Table 2. Input parameters

Parameter	Default Value
# of nodes	100
# of clusters	5
# of queries	200
# of peers per cluster	20
# of documents per peer	10
# of path expressions in description	4
K	0.5
Size of document	2KB-8KB

6. Performance Evaluation

To evaluate the performance of our approach we simulated a network of nodes and measured the performance of query routing when applying our clustering algorithm. Each node in the network stores 10 XML documents. For the generation of the XML documents we use the ToXgene generator [22]. We provide to the generator as input 5 different templates which produce documents belonging to 5 categories. From the 10 documents of each peer, 7 belong to a single category and the remaining 3 are selected uniformly at random. We fix the number of superpeers and therefore, clusters in the system to 5, one for each of the document categories.

We compared our approach of workload-aware clustering with an approach that creates a clustered overlay network based solely on the content of the nodes without exploiting any workload knowledge. Thus, the content-based approach includes in the cluster description only the path expressions that appear more frequently in the nodes content. Finally, we also used an approach that selects the superpeer each node attaches to at random and does not build any cluster descriptions at all.

We used a network of 100 nodes and fixed the number of peers per cluster to 20. As our performance metric we used the number of hops required for the query routing protocol to locate a specified number of results. We measure only the hops required for locating a node and not the hops required for returning the results to the node that issued the query. Furthermore, once a query reaches a cluster it is forwarded in parallel to all the nodes that belong in the cluster even if

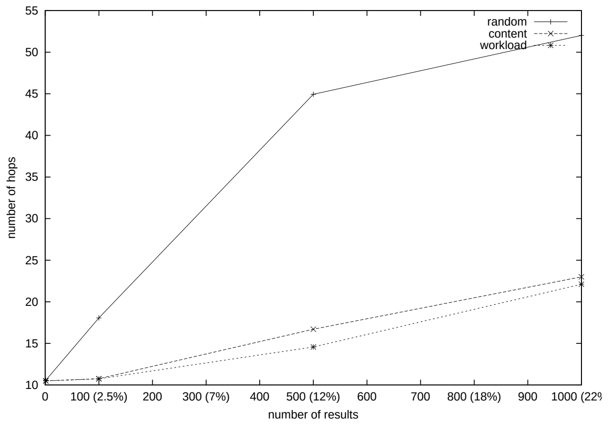


Figure 3. Routing performance with $T_1(QW)$

the results that are attained exceed the number of specified results requested.

We measure the performance of query routing while the system evolves to demonstrate how our approach works in a dynamic environment with changing conditions. In particular, we simulated batches of nodes joining the system interleaved with batches of queries execution. After each of the join batches the content-based approach updates its description while the workload-aware approach updates the cluster descriptions after a batch of peer join and a batch of queries is completed.

We conducted three sets of experiments by using two types of query workload with different characteristics for each of them. In the first type of query workload ($T_1(QW)$), the queries follow the same distribution as the data that belong to each peer. Queries for path expressions that appear often in the data of the peers appear more often in the query workload. This type of workload favors the content-based approach, since this approach includes into the cluster description the path expressions that appear more frequently in the nodes data. As most queries involve frequent data the content-based descriptions are able to determine efficiently which clusters are relevant to most queries. The second type of query workload ($T_2(QW)$) consists mainly of queries for data items that not appear very frequently in the nodes content. This type of workload clearly favors the workload-aware approach because as more queries for unpopular data become more frequent they are included into the appropriate cluster descriptions making the routing of the next queries for this data more efficient. The content-based approach has no means of adapting to the demands of the query workload and thus the request for unpopular data cannot be aided with the use of the cluster description.

Experiment 1. In this experiment, we fixed the number of path expressions in the cluster description to 4, which is about 6% of all the distinct path expressions that are contained in a cluster table, and the percentage K of expressions extracted from the query workload to 0.5.

Figure 3 illustrates our results when using $T_1(QW)$. The numbers in the x axis show the requested number of results and the percentage in the parenthesis denotes what percentage of the total number of available results this number represents. Obviously, the random approach has the worst overall performance. To locate the requested number of results it requires almost twice as much hops as the other two approaches. What is particularly interesting, is that the workload-aware approach performs slightly better than the content-based approach even for this type of workload that obviously favors the content-based approach.

As shown in Fig. 4 when using $T_2(QW)$, the workload-aware approach outperforms both the content-based and the random approach. We can see that for a number of requested results below 200, the difference in the number of hops between the content-based and the workload-aware method is very significant, the content-based method additionally visits about 10% of the network nodes. After the number of results becomes large enough, more than 200, both approaches visit about the same number of nodes, but still outperform the random method. This is the case when the number of results in a single cluster do not suffice to attain the requested number of results.

Experiment 2. In this experiment, we fixed the number of path expressions in the cluster description to 1, and compared the content-based approach, where 0% of the workload is included into the description (workload - 0%), with the workload-aware approach where 100% of the path expressions in the description are derived from the query workload (workload - 100%). This two cases represent the two extremes, either using only content or only workload information into the cluster description. We conducted the same experiment using $T_1(QW)$ (Fig. 5) and $T_2(QW)$ (Fig. 6).

When using $T_1(QW)$, we see that by using just the workload we still have slightly better performance from the content-based approach. The performance is similar to the case where we included into the description 50% from the query workload and 50% from the peers content. This is extremely interesting, because using such a description does not require having any knowledge on the content of the peers in the system. Thus, it can be applied even if the superpeer does not construct the path-count table for its cluster.

When we use $T_2(QW)$, the performance of the workload-based approach outperforms the content-based approach until the number of requested results can no longer be attained within a single cluster. The rest of the clusters are visited also and their description is updated to show that they contain the requested item. In future queries, the rout-

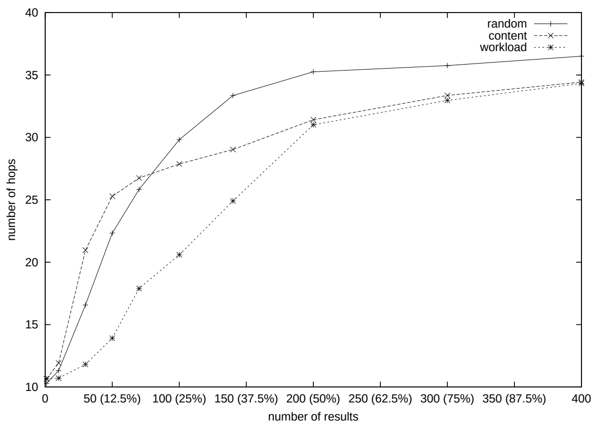


Figure 4. Routing performance with $T_2(QW)$

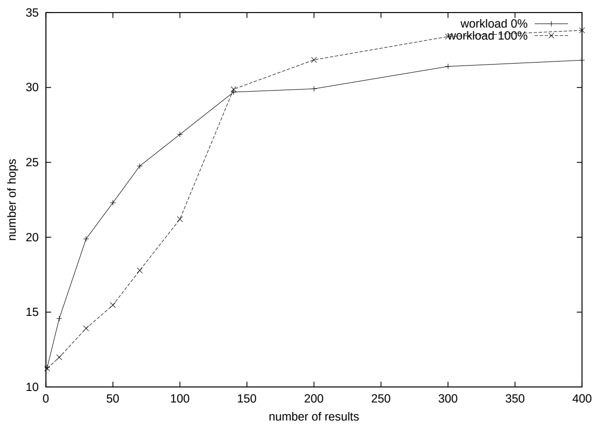


Figure 6. Workload vs Content using $T_2(QW)$

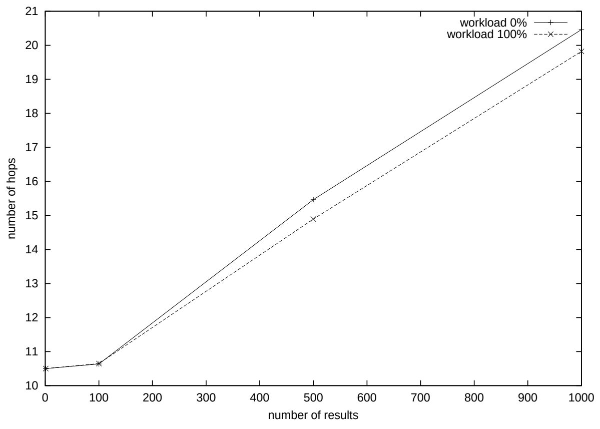


Figure 5. Workload vs Content using $T_1(QW)$

ing protocol may falsely try to satisfy a query from these clusters that only contain a limited number of results. Thus, the performance of the workload-aware approach deteriorates.

Between the two extremes we can further investigate the appropriate percentage of query workload information that is included into the cluster description. However, with the data set we used varying this percentage does not influence the behavior of the workload-aware approach significantly. We plan to further investigate this parameter by using other data sets and types of query workload.

Experiment 3. In this last experiment, we fixed K to 0.5 and examined how the number of path expressions that are included into a cluster description both for the workload-aware and the content-based approach affect the routing performance. Although one's intuition is that the larger the cluster description the better the performance of both the methods, we observed that this does not hold. Smaller descriptions with about 5% to 10% of all the path expressions in a cluster showed better performance than descriptions with 50% of the path expressions, for both types of the query workload. This is due to the fact that if we include too many paths in the description it seizes to be descriptive of the cluster and becomes too general, allowing all the queries to be routed to it. Thus, there is an optimal number of path expressions that should be included into the description. We plan to investigate how this optimal number can be dynamically determined, by conducting more experiments with different data sets and types of query workload.

7. Conclusions and Future Work

In this paper, we presented an approach for improving the query routing performance in a p2p system in which peers share data and resources described in XML format. Our approach is based on building clusters of peers that share similar data, by following a distributed procedure that requires no global knowledge of the shared data distribution. The clustering procedure is based on the use of rules that associate the existence of simple path expression patterns in a peer's data, to its likelihood of belonging to a specific cluster. The patterns that form the rules are selected from both the most popular path expressions in the peers content and the most popular queries in the query workload of the system. Furthermore, we described how these rules (i.e. the patterns that form them) change over time to reflect changes in the query workload and the content of the peers. Our experimental results show our workload-aware clustering procedure outperforms a strictly content-based approach, especially in the case where the query workload of the system mostly concerns data items that do not appear frequently in the peers content.

We plan to extend our approach to an adaptive clustering procedure that will not only change the clustering rules over time, but will also change the number of the clusters in the system, either by merging similar clusters, or by creating new clusters. We also wish to consider load-balancing parameters while changing the number of clusters. We also plan to use more sophisticated data structures to summarize the contents of each peer and cluster so as to minimize the space overhead. Finally, we want to see how our clustering procedure can be applied to p2p systems that use other overlay architectures and in particular, DHT-based p2p systems that support more efficient lookups.

8. Acknowledgements

Work supported in part by the GSRT Greece-Cyprus project SEMANET and the IP IST-15964 project AEOLUS.

References

[1] A. Aboulnaga, A.R. Alameldeen and J.F. Naughton. Estimating the Selectivity of XML Path Expressions for Internet Scale Applications. VLDB, 2001.

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

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

[4] A. Crespo and H. Garcia-Molina. Semantic Overlay Networks for P2P Systems. Computer Science Department, Stanford University. October 2002.

[5] C. Fellbaum. WordNet: An Electronic Lexical Database. MIT Press, 1998.

[6] Knowbuddy's Gnutella FAQ. http://www.rixsoft.com/Knowbuddy/gnutellafaq.html.

[7] G. Kokkinidis and V. Christophides. Semantic Query Routing and Processing in P2P Database Systems: ICS-FORTH. 1st Int. Workshop P2P and DB, 2004.

[8] G. Koloniari and E. Pitoura. Content-Based Routing of Path Queries in Peer-to-Peer Systems. EDBT, 2004.

[9] M.L. Lee, L.H. Yang, W. Hsu and X. Yang. XClust: Clustering XML Schemas for Effective Integration. CIKM, 2002.

[10] W. Lian, D.W. Cheung, N. Mamoulis and SM. Yiu. An Efficient and Scalable Algorithm for Clustering XML Documents by Structure. IEEE TKDE, Vol. 16(1), 2004.

[11] A. Loser, W. Siberski, M. Wolpers and W. Nejdl. Information Integration in Schema-Based Peer-To-Peer Networks. CAiSE, 2003.

[12] G. Miller. WordNet: A Lexical Database for English. CACM, Vol. 38(11), November 1995.

[13] Napster. http://www.napster.com/

[14] V. Papadimos, D. Maier and K. Tufte. Distributed Query Processing and Catalogs for Peer-to-Peer Systems. CIDR, 2003.

[15] N. Polyzotis and M. Garofalakis. Structure and Value Synopses for XML Data Graphs. VLDB, 2002.

[16] N. Polyzotis, M. Garofalakis and Y. Ioannidis. Approximate XML Query Answers. SIGMOD, 2004.

[17] S. Ratnasamy, P. Francis, M. Handley, R. Karp and S. Shenker. A Scalable Content-Addressable Network. SIGCOMM, 2001.

[18] C. Sartiani, P. Manghi, G. Ghelli and G. Conforti. XPeer: A Self-organizing XML P2P Database System. 1st Int. Workshop P2P and DB, 2004.

[19] M. Schlosser, M. Sintek, S. Decker and W. Nejdl. A Scalable and Ontology-Based P2P Infrastructure for Semantic Web Services. 2nd Int. Conf. on Peer-to-Peer Computing, 2002.

[20] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. SIGCOMM, 2001.

[21] M. Theobald, R. Schenkel and G. Weikum. Exploiting Structure, Annotation, and Ontological Knowledge for Automatic Classification of XML Data. WebDB, 2003.

[22] http://www.cs.toronto.edu/tox/toxgene/

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

[24] World Wide Web Consortium. Extensible Markup Language (XML) 1.0 (Second Edition). http://www.w3.org/TR/REC-xml. October 2000.

[25] M.J. Zaki and C. C. Aggarwal. XRules: An Effective Structural Classifier for XML Data. SIGKDD, 2003.