An XML Routing Synopsis for Unstructured P2P Networks

Qiang Wang
University of Waterloo
q6wang@uwaterloo.ca

Abhay Kumar Jha*
IIT, Bombay
abhaykj@cse.iitb.ac.in

M. Tamer Özsu
University of Waterloo
tozsu@uwaterloo.ca

Abstract

Many emerging applications that use XML are distributed, usually over large peer-to-peer (P2P) networks on the Internet. The deployment of an XML query shipping system over P2P networks requires a specialized synopsis to capture XML data in routing tables. In this paper, we propose a novel graph-structured routing synopsis, called kd-synopsis, for deployment over unstructured super-peer based P2P networks. This synopsis is based on length-constrained FBsimulation relationship, which allows the balancing of the precision and size of the synopsis according to different space constraints on peers with heterogeneous capacity. We report comprehensive experiments to demonstrate the effectiveness of the kd-synopsis.

1 Introduction

In recent years, Peer-to-Peer (P2P) architecture has become a popular decentralized platform for many Internet-scale applications such as file-sharing[1], instant messaging[2], and computing resource sharing[3]. Meanwhile, XML data are increasingly used as a format for data exchange and storage on the Internet, such as XML-based sensor data [12] and Web service data defined in WSDL and SOAP. Thus, it is increasingly important to process queries efficiently over data deployed in large-scale P2P networks, where centralized catalogs are not always available.

Distributed (XML) query processing can follow either data-shipping or query-shipping approaches. Data shipping moves data to a query processing site, while query shipping systems route queries to where the data are located for processing. Query shipping is preferable in P2P systems because it can exploit the computational power of peers to process queries in parallel, and the cost of sending queries is usually much smaller than that of sending data.

Query shipping strategies vary based on different P2P overlay network architectures and routing protocols. In structured P2P architectures, data are placed on peers that are organized in a structured overlay network by using distributed hashing, and each peer manages, along with data, hash-based routing information for "neighboring" peers. In contrast, unstructured P2P architectures keep data on original peers and route queries either by flooding or by using super-peers. While flooding is effective when searching popular data with a lot of replicas [20], a super-peer based architecture makes routing more efficient by organizing peers hierarchically where upper-level peers maintain information about the content of their lower-level ones and use this for routing. Such an architecture combines the advantages from both centralized systems (e.g., the exploitation of heterogeneity of peers) and distributed systems (e.g., scalability and robustness), and has been employed in practical P2P file-sharing systems [3].

In this paper, we focus on XML query routing in query-shipping P2P systems under super-peer based overlay network architectures. More specifically, we propose an effective XML synopsis that can be used for maintaining routing state. For the sake of simplicity, we assume that each peer contains one tree-structured XML document (i.e., without ID or IDREF). In the case that a peer has multiple documents, we can model them under a virtual root. We are concerned with a commonly used subset of XPath [9], called Branching Path Query (BPQ) [16]. Briefly, BPQ covers *self*, *child*, *descendant*, *descendant-or-self*, *parent*, *ancestor* and *ancestor-or-self* axes. Moreover, we consider the conjunction (i.e., *and* operator) of predicates. Since we do not consider *negation* in queries, we actually work on a subclass of BPQ, defined as BPQ$^+$ [24].

Our solution, called *kd-synopsis*, is a graph-structured synopsis over XML data that can easily balance its precision and size. Here the k and d are

*Work done while the author was visiting the University of Waterloo.

[1]Gnutella: http://www.gnutella.com

[2]Skype: http://www.skype.com

[3]Seti@home: http://setiathome.ssl.berkeley.edu

length constraints to be imposed over the backward and forward-simulation relationships (Section 2), which are theoretical foundations of our routing synopsis. Briefly, the contributions of this paper consist of three parts: First, to the best of our knowledge, we give the first formal definition of a length-constrained FBsimulation relationship (Section 2) over XML data, which we call *kd-simulation*. Second, we develop a novel size-adjustable routing synopsis, called *kd-synopsis*. Finally, we address the aggregation and update maintenance issues for routing tables consisting of kd-synopses, and demonstrate the effectiveness of our design with extensive experiments.

The organization of the paper is as follows. We introduce the related work and background in Section 2. In Section 3 we precisely define the kd-simulation relationship and address the generation of kd-synopses over XML documents. Section 4 focuses on the XML query shipping enforced in super-peer based unstructured P2P networks. Experimental results are presented in Section 5, demonstrating the effectiveness of our routing synopsis. We conclude in Section 6.

2 Related Work and Background

2.1 Related Work

XML query shipping problem is addressed in both structured [14, 11, 13] and unstructured P2P [18] domains. Since we are interested in unstructured P2P systems in this paper, Koloniari and Pitoura's work [18] is the most relevant, where Bloom filters are used to capture the set of elements on document tree levels and paths with same lengths. The basic idea is to encode the set of all the elements on each document level into a specific Bloom filter (the Breadth Bloom filter), and encode the set of all the linear path strings with a fixed length into a specific Bloom filter (the Depth Bloom filter). Although Bloom filter is space efficient in checking membership over a set of elements, it has several disadvantages when used to encode hierarchically-rich XML data. First, it cannot precisely encode the ancestor-descendant relationship among elements, because Breadth Bloom filter does not capture the parent-child relationship among nodes at different levels, and Depth Bloom filter only captures linear XML paths with specific lengths by uniform hashing. Thus a query with ancestor-descendant axis cannot be checked against the data effectively. Second, to guarantee that there are no false negatives, which is desirable in query shipping systems, each document level has to correspond to a Breadth Bloom filter, so the size of the synopsis increases sharply when used to encode deep-structured documents, such as those following a recursive schema [19]. Finally, each peer in the system needs to use the same hash functions and agree on the size of the Bloom filters (i.e., size of bit sequences of the Bloom filter) corresponding to a specific level or a fixed path length. This may not be appropriate in large heterogeneous P2P systems. In this work, we develop a graph-structured synopsis to represent routing state, which can capture the XML hierarchial information more easily and is size-adjustable so as to fit the heterogeneous capacity on different peers.

Compact graph-structured synopses have been developed to index XML data. Milo et al. [22] employ forward-Bisimulation equivalence relationship (described in Section 2.2) to devise a reduced index for XML data with respect to linear path XML queries. Polyzotis et al. [23] develop XSketch synopsis for selectivity estimation, which also exploits the Bisimulation relationship to reduce the common structures in XML data. Further, Ramanan [24] proves that FBsimulation quotient is the smallest covering index[4] of XML data with respect to BPQ$^+$ queries, and the quotient can be much smaller than Bisimulation-based synopses. Kd-synopsis shares the graph structure of these synopses and builds on the same foundation of the FBsimulation relationship, but it can be significantly smaller than FBsimulation quotient. Moreover, to make the routing synopsis size-adjustable according to the heterogeneous capacity on different peers, we impose length constraints over the FBsimulation relationship such that we can easily balance the size and the precision through the constraint parameters. Kaushik et al. [17] also consider a length constrained equivalence relationship (i.e., k-bisimilarity) in their A(k)-index for XML indexing, but they impose length constraints only on the backward direction of the Bisimulation relationship, such that document nodes with different subtree structures can not be distinguished, leading to a less precise synopsis for BPQ$^+$ queries. Recently, Zhang et al. [27] propose Xseed, a new XML synopsis that uses an enhanced label-splitting graph. An evaluation of kd-synopsis against it will be an interesting future work.

2.2 Background

FBsimulation relationship has been shown to capture similar structures in XML data for BPQ$^+$ queries [24]. We use it as the theoretical foundation of kd-synopsis to remove redundancy with respect to the positive check[5] of BPQ$^+$ queries. FBsimulation is

[4] An index DI of document D is a covering index with respect to a query set if no query in the set can distinguish between two nodes of D that correspond to the same indexing node in DI.

[5] The check of whether the result set of evaluating a query against a document (or a synopsis) is non-empty.

an extensively researched concept, which is used to describe the structure of semi-structured data [8], and has recently been employed to build compact covering indexes for XML data [24]. Given a directed graph $G = (V, E)$, where V is the node set and E is the edge set, for a node $v \in V$, let $label(v)$ return the label of v. Then

- backward-simulation ($\ll_B$) is a binary relation over V^2, such that, $\forall u, v \in V$, $u \ll_B v$ iff
  - $label(u) = label(v)$;
  - for every parent node u' of u in G, there is a parent v' of v such that $u' \ll_B v'$.
- forward-simulation ($\ll_F$) is a binary relation over V^2, such that, $\forall u, v \in V$,$u \ll_F v$ iff
  - $label(u) = label(v)$;
  - for every child node u' of u, there is a child v' of v such that $u' \ll_F v'$.
- FBsimulation is a binary relation ($\ll_{FB}$) over V^2, such that $\forall\ u, v \in V$, $u \ll_{FB} v$ (called u is FBsimulated to v) iff
  - $label(u) = label(v)$;
  - for every child node u' of u, there is a child v' of v such that $u' \ll_{FB} v'$;
  - for every parent node u' of u, there is a parent v' of v such that $u' \ll_{FB} v'$.

For example, consider the document tree shown in the Figure 1, where we use subscripts to distinguish nodes with same labels for demonstration (i.e., c_1, c_2, c_3 all have label c). While t_4 is FBsimulated to t_2, it is not FBsimulated to t_1 because they do not share matching incoming paths. Similarly, s_2 is FBsimulated to s_3 , while it is not FBsimulated to s_4 because they have distinct subtrees. An interesting observation is that, once a node is FBsimulated to other nodes, it becomes redundant when used in the positive check of BPQ$^+$ queries against documents. For example, given a query */b/c/s*, we can correctly determine that the query is positive to the document in Figure 1 only if s_2 (or s_3, s_4) exists, which means we can collapse s_1 into s_2 (or s_3, s_4) without affecting the effectiveness of positive check for query routing. We will exploit this finding in developing kd-synopsis. Note that we are not using the FBsimulation quotient synopsis discussed in [24], where two nodes are allowed to be collapsed into each other if and only if they are mutually FBsimulated. Thus, regarding the same example, s_1 can not be collapsed because none of the other nodes with common label s are FBsimulated to it. So our routing synopsis will be more compact than FBsimulation quotient.

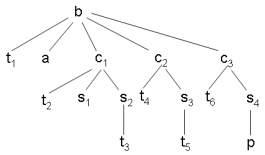


Figure 1. An XML document tree

3 kd-synopsis

An XML routing synopsis should satisfy two properties to be effective in P2P environments. First, it should be as precise as possible such that the routing synopsis can discriminate queries accurately – this will save the communication cost since false positive queries[6] will not be routed. Second, the routing synopsis should adapt to heterogeneous capacities (i.e., storage space for routing information) on different peers. In this Section, we describe kd-synopsis, which is a novel routing synopsis based on kd-simulation, a length-constrained version of FBsimulation relationship. Through the adjustment of length constraints over kd-simulation, kd-synopsis can obtain a level of precision with respect to specific size constraints.

3.1 Definition of kd-simulation

In this section, we define kd-simulation relationship that can identify XML element nodes that have similar structures within the specified incoming and outgoing levels. Specifically, based on the notions of backward-simulation, forward-simulation, and FBsimulation, we define their length-constrained variants respectively: *k-backward-simulation*, *d-forward-simulation* and *kd-simulation*.

- Given a graph $G = (V, E)$, **k-backward-simulation** ($\ll_B^k$) is a binary relation over V^2 defined inductively as follows ($k \geq 0$). Given $u, v \in V$,
  - $u \ll_B^0 v$ iff $label(u) = label(v)$;
  - $u \ll_B^k v$ iff (1) $label(u) = label(v)$; (2) for every parent node u' of u, there is a parent v' of v such that $u' \ll_B^{k-1} v'$.
- Given a graph $G = (V, E)$, **d-forward-simulation** ($\ll_F^d$) is a binary relation over V^2 defined inductively as follows ($d \geq 0$). Given $u, v \in V$,

[6] A query is false positive when it is determined as positive to the synopsis while it is actually negative to the data on which the synopsis is generated.

– $u \ll_F^0 v$ iff $label(u) = label(v)$;
– $u \ll_F^d v$ iff (1) $label(u) = label(v)$; (2) for every child node u' of u, there is a child v' of v such that $u' \ll_F^{d-1} v'$.

Briefly, k-backward-simulation identifies similar incoming paths within k XML tree levels, while d-forward-simulation identifies similar outgoing structures within d XML tree levels. For example, in the XML document tree shown in Figure 1, $t_1 \ll_B^0 t_2$ because they share the same label, but $t_1 \not\ll_B^1 t_2$ because they have different parents (i.e., different incoming edge one level away); $c_2 \ll_F^1 c_3$ but $c_2 \not\ll_F^2 c_3$ because they have distinct outgoing edges two levels away (i.e., *s/t* and *s/p* respectively).

Now we define kd-simulation as a reflexive and transitive relation over V^2, with k and d values as the length constraints over backward and forward-simulation relationship respectively.

- Given a graph $G = (V, E)$, **kd-simulation** ($u \ll^{(k)(d)} v$, called u is kd-simulated to v) is a binary relation over V^2 defined inductively as follows. Given $u, v \in V$,
  – $u \ll^{(0)(d)} v$ iff $label(u) = label(v)$ and $u \ll_F^d v$;
  – $u \ll^{(k)(0)} v$ iff $label(u) = label(v)$ and $u \ll_B^k v$;
  – $u \ll^{(k)(d)} v$ iff (1) $label(u) = label(v)$; (2) for every child node u' of u, there is a child v' of v such that $u' \ll^{(k)(d-1)} v'$; (3) for every parent node u' of u, there is a parent v' of v such that $u' \ll^{(k-1)(d)} v'$.

For example, in Figure 1, t_4 is kd-simulated to t_6 with respect to $k = 1, d = 1$ because their parent nodes $c_2 \ll^{(0)(1)} c_3$, and neither of them has children nodes. However, t_4 is not kd-simulated to t_6 anymore under $k = 1$ and $d = 2$, since $c_2 \not\ll^{(0)(2)} c_3$ (i.e., $c_2 \not\ll_F^2 c_3$). Note that when both k and d are equal to G's graph diameter (i.e., the tree depth when G is an XML document tree), kd-simulation evolves into the FBsimulation relationship, which is the basis of the covering index graph for BPQ$^+$ [24], and when both k and d are equal to zero, it degrades into the label-equivalence relationship, which is the basis of the label-splitting graph[7] that captures all the basic label and edge information in G.

Efficient algorithms have been designed to compute the forward-simulation relationship among graph nodes [10, 15]. We propose an algorithm (Algorithm 1) to cover our special circumstances. This algorithm is similar to that given by Henzinger et al. [15], but has two important differences. First, besides forward-simulation, we also consider backward-simulation. Second, we apply length constraints over the FBsimulation relationship. The implementation is straightforward, containing iterations for the computation of d-forward-simulation (k-backward-simulation), where in each iteration, the distinction among nodes incurred by different outgoing edges (incoming edges) are propagated to an upper (lower) level in the tree. Due to space limitation, we leave the correctness proof of the algorithm to the long version of this paper [25]. Since Algorithm 1 is a constrained version of Henzinger et al. [15], the time complexity is still $O(|V||E|)$ with respect to the graph G.

Algorithm 1 *compute_kd_simulation*

```
Input: G = (V, E), k, d
Output: for each node v ∈ V, the set of vertices sim(v) such that
sim(v) contains all the nodes v satisfying v ≪^(k)(d) u
Auxiliary functions: ∀x, y ∈ V, pre(y) = {x|(x, y) ∈ E},
post(x) = {y|(x, y) ∈ E}

for v ∈ V do
  sim(v) ← the set of the vertices with the same label as v's;
end for

k_tmp ← k; d_tmp ← d;

//(d)-forward-simulation
for all v ∈ V do
  sim'(v) ← sim(v);
end for
while d_tmp > 0 do
  for all v ∈ V, u ∈ sim(v) do
    if ∃v_c ∈ post(v), such that there does not exist u_c ∈ sim(v_c),
    where u_c ∈ post(u) then
      remove u from sim'(v);
    end if
    d_tmp ← d_tmp − 1;
    for all v ∈ V do
      sim(v) ← sim'(v);
    end for
  end for
end while

//(k)-backward-simulation
for all v ∈ V do
  sim'(v) ← sim(v);
end for
while k_tmp > 0 do
  for all v ∈ V, u ∈ sim(v) do
    if ∃v_p ∈ pre(v), such that there does not exist u_p ∈ sim(v_p),
    where u_p ∈ pre(u) then
      remove u from sim'(v);
    end if
    k_tmp ← k_tmp − 1;
    for all v ∈ V do
      sim(v) ← sim'(v);
    end for
  end for
end while
```

3.2 Building kd-synopsis

Algorithm 1 determines the kd-simulation relationship among document nodes, based on which we

[7] In such a graph, nodes are merged if they have common labels, and edges are merged if they have common labels on both ends.

collapse an arbitrary graph node u to v ($u \neq v$) if $u \ll^{(k)(d)} v$. This process continues until no node remains that can be kd-simulated by any others. Accordingly, an arbitrary edge $e_1 = (u_1, v_1)$ is collapsed into $e_2 = (u_2, v_2)$ if both $u_1 \ll^{(k)(d)} v_1$ and $u_2 \ll^{(k)(d)} v_2$. Additionally, since absolute BPQ$^+$ queries (i.e., queries starting from root level) require the synopsis to capture level information of the original document, we track where the root node of the original document is collapsed into. The algorithm is straightforward, and details can be found in [25]. For demonstration, Figure 2 depicts the kd-synopses of the example document under different k and d values. Note that in the kd-synopsis with $k = 1$ and $d = 2$, c_3 is not collapsed because it has a distinct outgoing edge (i.e., s/p) two levels away.

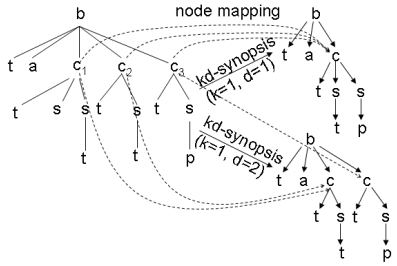


Figure 2. kd-synopsis

The size of the kd-synopsis is bounded by the size of the original document, and the reduction ratio depends on k and d values. The smaller k and d are, the higher the reduction ratio will be, and in the extreme case when $k = d = 0$, the kd-synopsis is the same as the label-splitting graph, which is the smallest synopsis that keeps original label and edge information.

4 kd-synopsis based Routing

kd-synopsis can be used in all kinds of routing-based P2P networks. In this work, we consider super-peer based unstructured P2P networks, where there are plain peers and super-peers, both of which keep XML data, but only the latter acts as routers for query shipping. Since peers are heterogeneous with respect to the storage space for the routing information, the routing states are organized under space constraints (Section 4.1). A query can be originated on an arbitrary peer. If the peer is a plain peer, the query will be forwarded to its super-peer, otherwise if the peer is a super-peer, based on the routing information stored there, the query will be routed to peers where the query can be potentially answered. In parallel, the query will also be forwarded to upper-level super-peers. For simplicity, we assume that top-level super-peers broadcast queries among each other such that no propagation of routing information happens among them. Since peers may join and leave the network dynamically, we will also discuss the update of the routing information (Section 4.2).

4.1 Routing State and Routing Decision Making

The routing tables on super-peers are managed in a bottom-up way. To establish a routing entry on super-peers for a plain peer, a kd-synopsis of the document stored on the plain peer will be generated and sent over to its super-peer. This synopsis can be obtained by setting the initial values of k and d as the depth of the document, and then decreasing the (k, d) values step-by-step until the size of the synopsis satisfies the space constraint imposed by the super-peer. Due to lack of space, the details are left to the long version of this paper [25]. To establish a routing entry for a super-peer on upper-level super-peers, we first need to aggregate the routing information of the lower-level super-peers, which constitutes a set of kd-synopses. Given a space constraint for this super-peer S (i.e., the space allowed to store the information of S on its upper-level super-peers), a naive way to aggregate is to merge the synopses into one kd-synopsis under the minimum k and d values among all the k and d values of the synopses in routing table of S, denoted as k_{min} and d_{min} respectively. However, it is obvious that the precision of the synopsis is limited by k_{min} and d_{min}, which is undesirable when either k_{min} or d_{min} is small. So, instead of merging all synopses into a single synopsis, we organize the kd-synopses by using a sequence, which keeps the precision of the original kd-synopses as much as possible. The reason of employing a sequence instead of a set is that we can easily track where a synopsis is aggregated. Specifically, given a set of kd-synopses, we first sort them in a decreasing lexicographic order over (k, d) value pair. Then we union the synopses with same (k, d) value pair, which leads to a synopsis sequence with a strict decreasing order over (k, d) value pair. Afterwards, if the size of this sequence still violates the specified space constraint, we start to reduce the size of the sequence by pairwise aggregating synopses: in each iteration, we choose a pair of kd-synopses S_1 (e.g., under $k = k_1$ and $d = d_1$) and S_2 (e.g., under $k = k_2$ and $d = d_2$) such that S_1 and S_2 are the first pair in the sequence that satisfies $k_1 = k_2$ and $d_1 \leq d_2$. S_1 and S_2 are then aggregated into S_{agg}, a kd-synopsis with $k = k_1$ and $d = min(d_1, d_2)$. If the pair of S_1 and S_2 does not exist, which indicates that all the kd-synopses in the sequence have different k values, we aggregate the first pair of kd-synopses in the sequence. This aggregation process continues until the space constraint is satisfied.

In this way, we can easily locate an original kd-synopsis in the aggregated sequence: it is either located at a synopsis with the same k and d values, or located at the foremost synopsis with the same k value but a lower d value, otherwise, the synopsis must be aggregated into the foremost synopsis (i.e., with a lower k value) of the sequence. This will facilitate the update maintenance of the routing information which requires the location of a synopsis already aggregated in routing tables on super-peers.

The routing decision making of a query q against a routing entry works in a straightforward way. When q is routed to a super-peer, it is checked against the kd-synopses in the sequence of each routing entry. If q is positive to any kd-synopsis, it will be routed to the peer corresponding to the routing entry, where the routing process continues in the same way; in contrast, if q is negative to all the kd-synopses in the routing entry, q will not be sent to the corresponding peer. The check of whether a query q is positive against a kd-synopsis can be reduced to the evaluation of the query over the synopsis. A FBsimulation-based query evaluation strategy addressed by Ramanan [24] can be directly applied.

4.2 Maintenance of Routing Information

In a super-peer based P2P network, the routing information (i.e., kd-synopsis) is propagated and replicated on multiple peers. So it needs to be updated when peers join or leave the network. Specifically, when a joining peer P is connected to its super-peer SP, it propagates its routing information (i.e., the kd-synopsis) to SP and then SP forwards changes over its own routing table to upper-level super-peers in an iterative way. When P leaves the overlay network, its corresponding routing information needs to be removed from all upper-level super-peers that contain P's information.

Since kd-synopses are aggregated together in routing tables, we need counter mechanisms to record the frequencies of common edges and vertices. Given a kd-synopsis S, the corresponding frequency-enabled synopsis is built by attaching to each edge e an integer counter that keeps the sum of the occurrences of common edges that collapse into e. Then the maintenance of the routing information becomes straightforward. When a peer P joins, it will send its kd-synopsis S to its direct super-peer SP, where a routing entry is established. Then S, as the change over SP's routing information, is sent to SP's direct super-peer SSP, followed by a call of the aggregation mechanism (Section 4.1) to guarantee that the routing entry still satisfies the space constraint. Then all the changes that have occurred over this routing entry, represented as removal and addition of specific kd-synopses, are sent to SSP's super-peers iteratively. When a peer P leaves, its corresponding routing entry is removed from SP's routing table. If P is a super-peer, we need to move its children and descendant peers under other super-peers. Since this work is focused on synopsis issues, we do not consider a complicated mechanism for super-peer selection, and just move the P's children peers up a level. Accordingly, the routing information corresponding to P's children peers is sent to SP, where a new space constraint will be imposed over all routing entries. Afterwards, all the changes over SP's routing information are sent to its super-peer SSP, where we take the routing entry corresponding to SP, remove and insert kd-synopses accordingly, and adjust the sequence according to the space constraint. We leave details to the long version of this paper [25].

5 Performance Evaluation

5.1 Testbed

We generate random XML data from a repository of XML documents with rich structures, including DBLP [1], TPC-H XML data [4], XMark [7], Treebank [5], and XBench [6, 26]. The size of the data that we populate on each peer is uniformly distributed between 2k and 20k bytes [8]. Because most of the XPath query generators do not consider backward axes (i.e., parent and ancestor axes), we develop a query generator to generate random BPQ$^+$ queries over the repository.

Since there is no widely deployed P2P XML query processing system, we developed a simulator that simulates a super-peer based hierarchical overlay network, where the space constraint on peers is uniformly distributed between 10k to 20k bytes. Specifically, we extract a fixed number of maximum spanning trees from transit-stub hierarchical graphs generator by GT-ITM [2], then connect roots of the trees as top-level super-peers, which will broadcast all queries among each other. Our simulation is conducted centrally on a 3G MHz PC with 1G memory. Since the Bloom-filter based strategy [18] is a closely related work, we re-implement it and use it as a comparison system in part of the experiments.

5.2 Precision and Size of kd-synopsis

In this experiment we try to demonstrate the precision that the Bloom-filter based synopsis [18] and kd-synopsis can provide vis-a-vis the space consumed by them. We use the metric of false-positive ratio to measure the precision. Given a document and a

[8] According to [21], the typical XML documents on the Web are small and the average document size is around 4KB.

set of queries[9] *negative* to the document, the false-positive ratio is computed by dividing the number of queries that are positive to the synopsis by the size of the query set. For this experiment we extract 250 small documents between 2k to 20k bytes from each document in the repository (e.g., DBLP, XMark) and a certain number (25 in our case) of negative queries are generated over each document. We note the average values including the size of FBSimulation quotient, the size and false-positive ratio of the Bloom-filter based synopsis [18], and kd-synopses under $k \leq 2$ and $d \leq 2$. Here we take FBsimulation quotient into account so as to show that kd-synopses can be significantly smaller. The experimental results are recorded in Table 1. It is obvious that with the increasing of k and d values, the false-positive ratio of the kd-synopsis decreases sharply, while the size of the synopsis keeps smaller than Bloom-filter based synopsis, which demonstrates the effectiveness of kd-synopsis in balancing the precision and size.

5.3 Query Shipping Performance

In this experiment, we use the number of hops for routing queries to their relevant data in the overlay network as the metric of the query shipping cost. We consider randomly generated networks up to 1000 peers, where each peer keeps one XML document (size between 2k and 20k bytes) randomly generated over the documents in the repository. To demonstrate the advantage of the kd-synopsis over Bloom-filter based synopsis, we deploy the two synopses as the routing synopsis separately over the same network, and use 100 queries to test the average shipping cost. We also compare the query shipping cost with the optimal shipping cost, defined as the smallest number of hops to be used to ship queries to all the peers that the queries are positive to. Since Bloom-filter based synopsis does not handle backward axes (i.e., parent and ancestor axes) in BPQ$^+$ queries, we run separate experiments for queries with and without backward axes, where we only compare performance of kd-synopsis and Bloom-filter based synopsis under the latter scenario. Figure 3(a) shows the comparison between the kd-synopsis and Bloom-filter based synopsis, which demonstrates that the shipping cost based on kd-synopsis is 53% smaller than that based on Bloom-filter based synopsis. Moreover, the shipping cost based on kd-synopsis keeps close to the optimal shipping cost, further demonstrating the effectiveness of the kd-synopsis. In Figure 3(b), we test the shipping cost for a different set of BPQ$^+$ queries that include backward axes. The result once again shows that the average cost is very close to the optimal shipping cost.

[9]For comparison purpose, in this experiment we consider a subset of BPQ$^+$ queries containing only forward axes, since Bloom-filter based synopsis can not handler backward axes.

6 Conclusion

We make an initial study of a novel graph-structured XML synopsis for distributed XML query processing over P2P network. The primary problem we are interested in is how to locate distributed XML data relevant to XML queries in a super-peer based unstructured P2P network. A key part of the problem is on the design of a routing synopsis that can capture the rich hierarchical information of XML data. Taking both the precision of routing synopses and the heterogeneity of P2P systems into account, we propose kd-synopsis, a novel synopsis that is built over length constrained FBsimulation relationship. Based on this routing synopsis, we address the issues on query routing and routing information maintenance. Experiments show that our scheme is much cheaper on query shipping than the comparison system proposed by Koloniari [18].

Since we are considering BPQ$^+$ query class, which is a subset of XPath language, we expect to extend our strategy to include more constructs such as IDREF, negation operator and sibling axes. To implement those, we need to capture order and ID/IDREF information from the data. These information must be encoded together with the basic graph-structure in a compact way so as to avoid a blowup of the synopsis size. Another problem we are working on is the optimization of queries to facilitate the check of the positive of queries against routing information.

Acknowledgement. Thanks to G. Koloniari and E. Pitoura for sharing their implementation details of the Bloom-filter based synopsis [18].

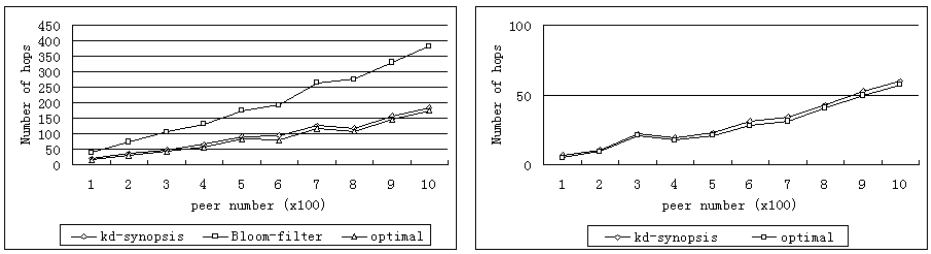
References

[1] DBLP. http://dblp.uni-trier.de/xml/.
[2] Georgia Tech Internetwork Topology Models. http://www.cc.gatech.edu/fac/Ellen.Zegura/graphs.html.
[3] KaZaA. http://www.kazaa.com.
[4] TPC-H. http://www.cs.washington.edu/research/xmldatasets/www/repository.html#tpc-h.
[5] Treebank. http://www.cis.upenn.edu/ treebank/.
[6] XBench. http://db.uwaterloo.ca/ ddbms/projects/xbench/.
[7] XMark. http://monetdb.cwi.nl/xml/index.html.
[8] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann, 1999.
[9] A. Berglund, S. Boag, D. Chamberlin, M. F. Fernandez, M. Kay, J. Robie, and J. Simeon. XML

	doc	FBQ	BF		kd (0, 0)		kd (0, 1)		kd (0, 2)		kd (1, 0)		kd (1, 1)		kd (1, 2)		kd (2, 0)		kd (2, 1)		kd (2, 2)	
	size	size	S	FP	S	FP	S	FP	S	FP	S	FP	S	FP	S	FP	S	FP	S	FP	S	FP
1	8612	1211	1609	0.99	286	1	569	0	569	0	318	0.997	875	0	876	0	318	0.997	875	0	876	0
2	8142	4246	2400	1	573	1	1465	0.103	1742	0.035	573	1	2796	0.075	3167	0	573	1	2910	0.075	3311	0
3	8293	3538	6840	0.758	669	0.79	1357	0.26	1716	0.115	1137	0.395	1965	0.193	2297	0.062	1787	0.315	3032	0.142	3282	0.031
4	10418	2348	4000	0.936	1158	1	1240	0.398	1244	0.398	1233	0.208	1378	0.001	1422	0.001	1294	0.001	1375	0.001	1507	0
5	11303	3871	2400	0.818	717	0.059	834	0	834	0	806	0.059	1233	0	1233	0	806	0.059	1233	0	1233	0

1: DBLP 2: TPC-H 3: Treebank 4: XBench(DCSD) 5: Xmark
FBQ: FBsimulaion quotient BF: Bloom-filter based synopsis S: size (in bytes) FP: False-positive ratio

Table 1. Precision and size



(a) (b)

Figure 3. Query shipping cost

path language (XPath) 2.0 W3C working draft. http://www.w3.org/TR/xpath20/, November 2003.

[10] B. Bloom and R. Paige. Transformational design and implementation of a new efficient solution to the ready simulation problem. *Science of Computer Programming*, 24:189–220, 1995.

[11] A. Bonifati, U. Matrangolo, A. Cuzzocrea, and M. Jain. XPath lookup queries in P2P networks. In *International Workshop on Web Information and Data Management*, 2004.

[12] A. Deshpande, S. K. Nath, P. B. Gibbons, and S. Seshan. Cache-and-query for wide area sensor databases. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 503–514, 2003.

[13] G. Erice, P. A. Felber, E. Biersack, G. Urvoy-Keller, and K.W.Ross. Data indexing in peer-to-peer DHT networks. 2004. Proc. 24rd Int. Conf. on Distributed Computing Systems.

[14] L. Galanis, Y. Wang, S. R. Jeffery, and D. J. DeWitt. Locating data sources in large distributed systems. In *Proc. 29th Int. Conf. on Very Large Data Bases*, pages 874–885, 2003.

[15] M. R. Henzinger, T. A. Henzinger, and P. W. Kopke. Computing simulations on finite and infinite graphs. In *Proc. IEEE Symp. Foundations of Computer Science*, pages 453–462, 1995.

[16] R. Kaushik, P. Bohannon, J. Naughton, and H. Korth. Covering indexes for branching path queries. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 2002.

[17] R. Kaushik, P. Shenoy, P. Bohannon, and E. Gudes. Exploiting local similarity for indexing paths in graph-structured data. In *Proc. Int. Conf. on Data Engineering*, pages 129–140, 2002.

[18] G. Koloniari and E. Pitoura. Content-based routing of path queries in peer-to-peer systems. In *Advances in Database Technology — EDBT'04*, pages 29–47, 2004.

[19] R. Krishnamurthy, V. T. Chakaravarthy, R. Kaushik, and J. F. Naughton. Recursive XML schemas, recursive XML queries, and relational storage: XML-to-SQL query translation. In *Proc. Int. Conf. on Data Engineering*, 2004.

[20] B. T. Loo, J. M. Hellerstein, R. Huebsch, S. Shenker, and I. Stoica. Enhancing P2P file-sharing with an Internet-scale query processor. In *Proc. 30th Int. Conf. on Very Large Data Bases*, pages 432–443, 2004.

[21] L. Mignet, D. Barbosa, and P. Veltri. The XML web: a first study. In *Proc. 12th Int. World Wide Web Conference*, pages 500–510, 2003.

[22] T. Milo and D. Suciu. Index structures for path expressions. In *Proc. 7th Int. Conf. on Database Theory*, 1999.

[23] N. Polyzotis and M. N. Garofalakis. Statistical synopses for graph-structured XML databases. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 358–369, 2002.

[24] P. Ramanan. Covering indexes for xml queries: Bisimulation - simulation = negation. In *Proc. 29th Int. Conf. on Very Large Data Bases*, pages 165–176, 2003.

[25] Q. Wang, A. K. Jha, and M. T. Özsu. A Simulation-based XML Routing Synopsis in Unstructured P2P Networks. Technical Report CS-2005-21, University of Waterloo, June 2005.

[26] B. B. Yao, M. T. Özsu, and N. Khandelwal. XBench benchmark and performance testing of XML DBMSs. In *Proc. 20th Int. Conf. on Data Engineering*, pages 621–633, 2004.

[27] N. Zhang, M. T. Özsu, A. Aboulnaga, and I. F. Ilyas. Xseed: Accurate and fast cardinality estimation for xpath queries. In *Proc. Int. Conf. on Data Engineering*, 2006.