BSI: BLOOM FILTER-BASED SEMANTIC INDEXING FOR UNSTRUCTURED P2P NETWORKS

Rasanjalee Dissanayaka[1] , Sushil K Prasad[2] , Shamkant B. Navathe[3] and Vikram Goyal[4]

[1]Department of Computer Science, College of Saint Benedict and Saint John's University, Collegeville, Minnesota USA
[2]Department of Computer Science, Georgia State University, Atlanta, Georgia USA
[3]College of Computing, Georgia Institute of Technology, Atlanta, Georgia USA
[4]IIIT-Delhi, New Delhi, India

ABSTRACT

Resource management and search is very important yet challenging in large-scale distributed systems like P2Pnetworks. Most existing P2P systems rely on indexing to efficiently route queries over the network. However, searches based on such indices face two key issues. First, majority of existing search schemes often rely on simply keyword based indices that can only support exact string based matches without taking into account the meaning of words. Second it is difficult, if not impossible, to devise query based indexing schemes that can represent all possible concept combinations without resulting in exponential index sizes.

To address these problems, we present BSI, a novel P2P indexing and query routing strategy to support semantic based content searches. The BSI indexing structure captures the semantic content of documents using a reference ontology. Our indexing scheme can efficiently handle multi-concept queries by maintaining summary level information for each individual concept and concept combinations using a novel space-efficient Two-level Semantic Bloom Filter(TSBF) data structure. By using TSBFs to represent a large document and query base, BSI significantly reduces the communication cost and storage cost of indices. Furthermore, We devise a low-overhead mechanism to allow peers to dynamically estimate the relevance strength of a peer for multi-concept queries with high accuracy solely based on TSBFs. We also propose a routing index compression mechanism to observe peers' dynamic storage limitations with minimal loss of information by exploiting a reference ontology structure. Based on the proposed index structure, we design a novel query routing algorithm that exploits semantic based information to route queries to semantically relevant peers. Performance evaluation demonstrates that our proposed approach can improve the search recall of unstructured P2P systems up to 383.71% while keeping the communication cost at a low level compared to state-of-art search mechanism OSQR [7].

KEYWORDS

Unstructured P2P Networks, P2P Search, Query Routing, Bloom Filters, Semantic Search

1. INTRODUCTION

P2P systems have become a popular means of sharing large amounts of data among users over recent years. They are considered an attractive solution for applications requiring high scalability, robustness and autonomy. Efficient resource discovery in basic unstructured P2P systems, however, suffers from high costs mainly due to lack of global knowledge. To make matters worse, with the advent of semantic web, more and more users associate their shared resources with semantic meta-information. This mandates that the P2P networks provide methods that allow a user not only to describe resources from a semantic viewpoint but also allow users to run more complex queries addressing several semantic properties or relationships among semantic entities and resources. Majority of search algorithms proposed for P2P networks to date, however, are simple keyword searches or title based searches. These are inadequate for formulating semantic based queries and consequently, do not provide semantically relevant results.

Index engineering is at the heart of P2P search methods. P2P indices can be broadly categorized to local, central and distributed indices. In a *local index*, a peer only keeps references to its own data. In a *central index* scheme such as one employed by Napster [1], a single server peer maintains data about all the other peers in the network in its index. Most widely used scheme is the *distributed index* scheme where peers maintain pointers towards targets in the network.

Current P2P indexing schemes face three main problems. First, overwhelming majority of indexing schemes proposed to date are simple keyword based indices. They do not associate semantics(i.e. meaning) to keywords in index resulting in poor search accuracy. Second, the handful of semantic based indexing schemes [11], [13] are either computationally intensive [13] or index size varies largely with the dimensionality of the content [13]. Third, maintaining query based indices (as opposed to simple keyword based indices)in a space efficient manner is challenging in P2P networks. Maintaining minimum routing state per peer is a crucial component in P2P searching. However, maintaining small size query based routing indices while not compromising the quality of information they hold is a challenging problem and has not been addressed sufficiently by the current body of work.

Maintaining query based indices in P2P networks are attractive for two reasons: First, such an index allows a peer to maintain more information about its neighborhood by allowing it to assign a relevance strength to a neighbor per query. Second, as indicate by many web search logs [19] majority of searches are multi-term(keyword or concept) queries that could benefit from query based indices to achieve better performance. Many of the proposed work, however, are optimized for single keyword queries. Handling multi-keyword queries is rather inefficient using single keyword indices. On the other hand, maintaining multi-term query based indices for better performance is simply impractical due to the resulting exponential size of indices. Little to no explicit measures have been taken to ensure high quality of indices regardless of the query length while maintaining small index sizes.

To address the aforementioned issues, in this paper we propose BSI, a semantics based indexing framework, which aims to improve the quality and efficiency of search in P2P networks by using a shared ontology as a reference for building index. Our indexing framework is designed for unstructured P2P systems where network imposes no structure on the overlay network or data placement. Our work addresses several important issues raised by current indexing schemes: First, our semantic based index framework takes into account the meaning of words thereby allowing peers to evaluate multi-concept queries. A reference ontology concepts serve as the index terms and relevance strengths for different queries (concept combinations)are maintained in the index. Each relevance strength is a combined measure of information content and the number of relevant documents reachable through a given peer and its neighborhood. Second, we maintain attractive small size index while maintaining the quality of index using Bloom filters. The size of the routing index is limited by the number of concepts in the ontology and can be easily compressed to accommodate space restrictions by utilizing hierarchical ancestor-descendant relations in semantic ontology. Finally, we explicitly capture the notion of multi-concept query based indexing in our index structure by allowing peers to maintain relevance strength for different queries for each neighbor. To maintain large volume of multi-concept queries and their corresponding relevance strength at a peer in a space efficient manner we introduce a novel Bloom filter based data structure called Two-level Semantic Bloom Filters(TSBF). TSBF is an extension to the traditional Bloom filter which incorporates the notions of ontology based meta data and relevance strengths of queries. TSBF allows us to limit the size of a routing index while maintaining large amount of summarized information regarding peers' strengths for possible queries to make a informed decision in routing multi-concept queries. Furthermore, We devise a low-overhead mechanism to allow peers to dynamically estimate the relevance strength of a neighbor for multi-term queries with high accuracy using TSBFs. We also propose a index compression mechanism to observe dynamic storage limitations of peers with minimal loss of information by exploiting the hierarchical relationships in the reference ontology structure. Finally, based on the proposed indexing scheme, we design a novel query routing algorithm that exploits semantic based information with different granularity to route queries to semantically relevant peers.

In BSI, we do not employ computationally expensive methods like LSI. Rather, we use concepts in a reference ontology to build peer local and routing indices to aid in the query routing process. The process of constructing peer indices is much simpler and consumes less memory and time in constructing them. Among many popular semantic indexing mechanisms OSQR [7], OLI [18], Ontsum [11] The related work most comparable to ours is OSQR [7], a semantic search protocol that maintains total relevant documents reachable through a neighbor for each concept from a reference ontology in its routing index as routing state. The authors propose a mechanism to calculate a conservative estimate of strength of a peer for a multi-concept query based on the number of documents reachable from that peer for each individual query concepts. The disadvantage of OSQR however is that its search cost is considerably high due its knowledge dissemination mechanism which utilize search messages for piggybacking data.

Overall, many existing work for semantic P2P search lacks the ability to efficiently answer multi-term queries with greater accuracy mainly due to lack of quality and quantity of routing state. BSI gracefully handles multi-term queries by exploiting both the concepts in query as well as structural relations of the query concepts in ontology to intelligently route a query to the most promising peer. Our solution is scalable as the index structures can easily be compressed to accommodate dynamic storage restrictions. Performance evaluation demonstrates that our proposed approach can improve the search efficiency of unstructured P2P systems while keeping the communication cost at a significantly lower level compared with state-of-art unstructured P2P systems. The rest of the paper is organized as follows: We present the related work in Section 2 . The preliminary concepts used in the paper are presented in Section 3. The P2P system architecture including routing index construction and maintenance and query routing are given in Section 4. In Section 5, we evaluate the proposed algorithms via simulation. Finally we conclude and summarize our work in Section 6.

2. RELATED WORK

Many search mechanisms for P2P networks have been proposed in the literature over the past few years. While some notable blind search mechanisms proposed for unstructured networks include flooding [2] and random walk [12], popular informed search methods include Routing Indices [21] and intelligent search [10]. However, these methods are intended for simple keyword searches only.

Semantic Overlay Networks (SON) [6] is one of the earliest works in incorporating semantic aspects into P2P file sharing applications. Here, authors propose a peer clustering approach based on semantic content a peer shares. This clustering methodology is completely distributed and also supports self-organization. However, in SON, peer joining process requires flooding the network to request classification hierarchy and also for finding semantically related peers. P2P systems such as Edutella [15] and InfoQuilt [3] use flooding as a basis for semantic search. Edutella proposes an RDF metadata model and is a super-peer based mechanism. However, the notion of super-peers makes the solution less attractive as super-peers need more resources than their sub-peers to maintain all routing indices as well as RDF triples representing sub-peers. This also leads to a substantial overhead in index updating at super-peers when a node joins, leaves, or changes its content. Compared to this, our approach is much simpler and involves minimal overhead in terms of local knowledge maintenance within peers. There is no notion of super peers and each peer only keeps a simple semantic based Bloom filter (TSBF) per its neighbor. pSearch [20] uses a semantic vector to describe and find the resources in the P2P network. SemreX [9] uses a concept tree to construct a semantic overlay network on top of P2P overlay. Both pSearch and SemreX use Latent Semantic Indexing (LSI) [5] to map documents to semantic vectors. LSI however is computationally expensive, and when the network content changes frequently, this method needs to be applied often. In BSI, we do not employ computationally expensive methods like LSI. Rather, we use concepts in a reference ontology to build peer local and routing indices to aid in the query routing process. The process of constructing peer indices is much simpler and consumes less memory and time in constructing them. OntIndex [13] is a ontology based indexing mechanism where authors propose a semantic based search algorithm for unstructured P2P systems. However, the notion of semantic information content is not taken into account in [13] resulting in low search performance. Also, index generation and updating in OntIndex is resource consuming and generates considerable traffic. OSQR [7] is a ontology based semantic search protocol which provides a rich semantic knowledge representation of peers by combining the both total documents reachable from a peer and the semantic richness of the peer based on the information content of the reachable documents. OSQR, however, heavily relies on knowledge dissemination mechanism which utilize search messages for piggybacking data contributing to increased search cost. OLI [18] is yet another ontology based indexing method for structured P2P systems. Semantic search methods applied in structured networks such as in OLI however, generally suffer from the shortcoming of structured overlays such as strict enforcement of data placement and high maintenance cost of peers joining, and leaving as well as the cost of content updating. To avoid these problems, we consider unstructured P2P overlay for our work. Ontsum [11] is another ontology based indexing scheme where heterogeneous ontologies between peers are assumed. This approach however requires computationally expensive ontology generation and ontology mapping techniques to be employed in a distributed setting and allow only RDQL based queries which requires some level of end user expertise to be used.

3. PRELIMINARIES

3.1. Network Topology

We consider an unstructured P2P network with a large set of peers *{P_1,P_2,..., P_P}*. Each peer in the network can only communicate with its direct neighbors in one hop distance. A peer is assumed to have on the average γ number of neighboring peers. Peers may leave and join the network at any time. Each peer has a local text document database that can be accessed through a local index. The peer uses its local index to evaluate all the content queries and returns pointers to the documents having the queried content.

3.2. Semantic Ontology

We assume the presence of a global reference semantic ontology which is known and agreed upon by every peer in the network. The ontology is a set of m semantic concepts $C=\{c_1,c_2,..., c_m\}$ which are related to each other via IS-A relationship (i.e., hypernym/ hyponym). We denote all the leaf concepts in the ontology by C_l .The root concept of the ontology is denoted by C_r.

3.3. Data Distribution

As stated earlier, each peer has a set of text documents. Each document is represented as a vector of concepts from C_l using a Vector Space Model, a standard technique for representing document contents in the area of information retrieval. Each document vector consists of a vector of concept relevancy scores. To construct a document vector, the document is subjected to a process of text mining followed by word sense disambiguation to identify those concepts that exist in the document along with their concept frequencies(i.e. number of occurrences of a concept in the document).Then the concept frequencies of each concept are normalized to a [0-1] range by applying maximum frequency normalization by dividing them by the maximum concept frequency for that concept encountered by the peer in its local document collection.

3.4. Semantic Query

A query consists of the conjunction or disjunction of one or more concepts from the leaf concepts (C_l) of the ontology. Existing techniques [4], [5] proposed by the research community can be employed to convert keyword queries to concept queries. A query returns a set of k documents having its relevancy above some user defined threshold value. The relevancy of a document for a given query is computed using concept vector similarity with respect to the query concepts. We use the well known cosine similarity measure to calculate the similarity between a query Q and document vector D:

$$Similarity(D,Q) = \frac{D.Q}{||D||.\,||Q||} \tag{1}$$

A document is said to *qualify* as an answer for a given query if the cosine similarity between its document vector and query exceeds a predefined relevance threshold.

3.5. Bloom Filters

A Bloom Filter (BF) is a data structure suitable for performing set membership queries very efficiently. A Standard Bloom Filter representing a set $S=\{s_1,s_2,..., s_n\}$ of n elements is generated by an array of m bits and uses k independent hash functions $h_1,h_2,...,h_k$. They are space efficient data structures which provide constant time lookups and no false negatives. The downsides of BFs are that they can result in false positives and do not allow item deletion. However, based on the application requirement the false positive rate can be significantly lowered. Therefore care must be taken in choosing k and m so that the false positive rate is acceptable. It has been shown that the probability of a false positive P_{false} can be given by the following equation

$$P_{false} = (1 - e^{-kn/m})^k \tag{2}$$

There are many variants of standard BF such as Spectral BF, counting BF, compressed BF, etc. In our work we utilize both standard Bloom filters and Spectral Bloom Filters (SBF). SBFs is an extension of the traditional BF for multi-sets allowing filtering of elements whose multiplicities are below a threshold. SBFs allow querying on item multiplicities as well as deletion. SBFs maintain a counter per bit in the bit array to keep track of the number of times the bit is set. Simply taking the minimum count over all bits set for a given element will produce the multiplicity of that element in the represented multi-set.

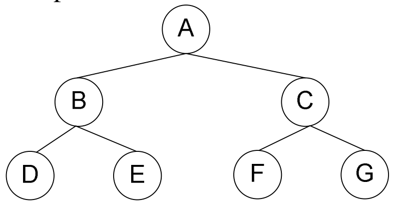


Figure 1. Ontology with seven concepts and IS-A relations between them.

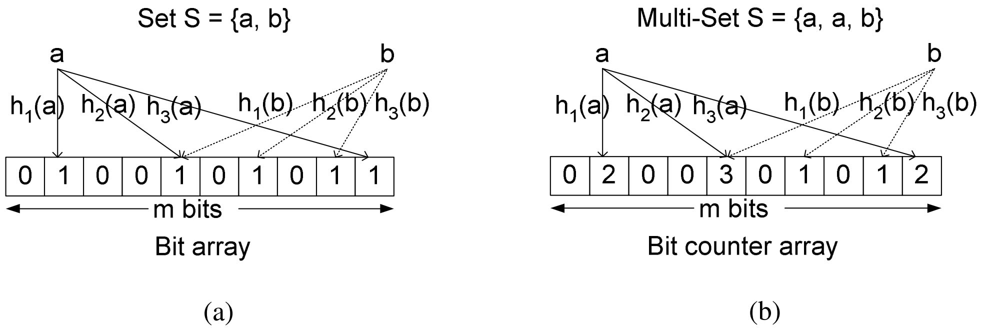


Figure 2. (a) A traditional Bloom filter with three hash functions. (b) A Spectral Bloom filter with three hash functions.

4. SYSTEM DESIGN

Three key issues in current indexing strategies are (i) their inability to associate semantic of represented content , (ii)low efficiency in routing multi-term queries and (ii)large index sizes. In order to overcome the shortcomings of existing indexing strategies and semantic query evaluation techniques, we put forward BSI(Bloom filter-based Semantic Indexing). We begin with the design of the Two-level semantic Bloom Filter (TSBF), a space efficient data structure that represents probabilistic information regarding contents a peer share with the network. We then describe our routing index structure used for efficient query routing.

4.1. Two-level Semantic Bloom Filter (TSBF)

Bloom filters serve as appropriate index structures for resource discovery due to their scalability, and low cost storage and distribution. However, they do not support semantic based multi-term queries as they have no means of representing probabilistic information regarding ontological data. To this end, we introduce a novel Bloom filter based data structure (TSBF) that aim at supporting efficient conjunctive (and disjunctive) semantic query routing in unstructured P2P networks.

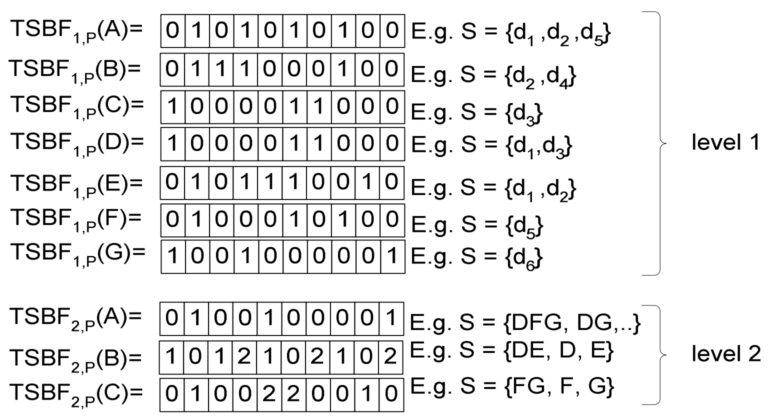
TSBF is an extension of the traditional Bloom Filter to encode probabilistic information regarding richness of a peer for different queries based on its local documents with the same false positive probability as the traditional Bloom filters. In particular, it encodes goodness of a peer for different multi-concept queries in terms of its distance to documents, the size of the document collection and information content of these documents. Posed with the query *"Is Query Q a member of TSBFP ?"* the TSBFP of peer P returns a relevance strength of P for Q based on information encoded in it. In our work, we define this relevance strength of a peer to be the total number of local documents in P for Q.

To implement this functionality TSBF is designed as a two-level Bloom filter where level 1 ($TSBF_{1,P}$) represents the set of qualified documents in P and level 2 represents the set of queries answerable by P. In particular, for level 1 there exist multiple bit arrays each representing the local documents of P rich in C ($TSBF_{1,P}(C)$). These bit arrays are similar to those employed in traditional Bloom filters and is supported by a sufficiently large body of research work [14], [16], [17] that allows us to estimate number of documents reachable for a multi-concept query solely based on these bit arrays.

Similar to level 1, level 2($TSBF_{2,P}$) also contains multiple bit arrays each representing different multi-concept queries that whose concepts have C as the least common ancestor in the ontology hierarchy for which P has at least one qualified document in its local document collection ($TSBF_{2,P}(C)$). To enable associating the number of relevant documents reachable through P for each query Q we implement each bit array in level 2 as Spectral Bloom Filters(SBFs). SBFs maintains a counter per each bit in a bit array thereby allowing deriving the number of times an element is added to bit array. In a level 2 bit array elements are queries and a given query Q is added as many times as the number of qualified documents exist in P for Q.

Intuitively, while level 1 contain a bit array per ontology concept, level 2 only needs to maintain a bit array per non-leaf ontology concept. Posed with a query Q, $TSBF_{2,P}$ returns the actual number of documents in P qualifies as answers for Q whereas the $TSBF_{1,P}$ provides an estimate of the same. While both provide two alternative ways for estimating the number of documents in P for Q, $TSBF_{2,P}$ is proffered over the other as it provides exact information regarding P's strength for a given query. When Q does not exist as a member in $TSBF_{2,P}$ then $TSBF_{1,P}$ is used to derive an estimate.

In our implementation, we size the set of bit arrays in both levels in a TSBF the same , but their sizes can be adjusted independently. Similar to traditional Bloom filters, TSBF uses a fixed number of hash functions $h_1,h_2,...,h_k$. Figure 3 shows a TSBF of a peer P for the ontology given in Figure 1. In the figure, the TSBF is a set of 10 bit arrays, 3 of which is for level 2, each representing possible queries per nn-leaf concept in the ontology while the other 7 is for level 1, where each bit array represents documents reachable for the given concept.



TSBF1,P(A)= 0 1 0 1 0 1 0 1 0 0 E.g. S = {d1 ,d2 ,d5}
TSBF1,P(B)= 0 1 1 1 0 0 0 1 0 0 E.g. S = {d2 ,d4}
TSBF1,P(C)= 1 0 0 0 0 1 1 0 0 0 E.g. S = {d3}
TSBF1,P(D)= 1 0 0 0 0 1 1 0 0 0 E.g. S = {d1,d3}
TSBF1,P(E)= 0 1 0 1 1 1 0 0 1 0 E.g. S = {d1 ,d2}
TSBF1,P(F)= 0 1 0 0 0 1 0 1 0 0 E.g. S = {d5}
TSBF1,P(G)= 1 0 0 1 0 0 0 0 0 1 E.g. S = {d6}
level 1
TSBF2,P(A)= 0 1 0 0 1 0 0 0 0 1 E.g. S = {DFG, DG,..}
TSBF2,P(B)= 1 0 1 2 1 0 2 1 0 2 E.g. S = {DE, D, E}
TSBF2,P(C)= 0 1 0 0 2 2 0 0 1 0 E.g. S = {FG, F, G}
level 2

Figure 3. The TSBF for a peer P with documents d_1, d_2, d_3, d_4, d_5, d_6. for ontology in Figure 1.

Such a TSBF of a peer serves two purposes: it allows fast query processing within the peer and also serve as the foundation for constructing peer's routing index. Use of Spectral Bloom filters in place of traditional Bloom filters allows associating a frequency of occurrence to each individual document/concept combination which adds to the information quality of Routing indices.

1) TSBF Construction: The TSBF of a peer P, $TSBF_P$ is constructed at start-up when peer joins the network for the first time. Initially all the bits of level 1 bit arrays are initialized to false while the all the bits in level 2 are initialized with a counter value zero. To populate level 1 bit arrays peer can analyze its document vectors to identify those qualifying documents per ontology concept and insert into the bit array responsible for the concept to populate them. In other words, P insert all its local documents that can answer concept C in its $TSBF_{1,P}(C)$ bit array.

Following the same process for populating level 2 bit arrays, however, leads to practical issues. We could ideally generate all possible concept combinations (queries)that exceeds a predefined cosine similarity threshold per local document of P and insert them in the appropriate bit arrays in level 2. Note that, same concept combination can be answered successfully by multiple documents thus allowing the combination to be inserted into the relevant bit array that many times. For instance, a concept combination answerable by a document d in P is inserted into

$TSBF_{2,P}$ *(C)* where C is the least common ancestor of concepts in combination based on the reference ontology. This method of level 2 TSBF population is certainly a viable option for those peers with sufficiently large amount of storage at hand as the number of concept combination that can be answerable by a peers document collection could be quite large. Such a technique is often unfavorable, however, as this might lead to high false positive rates due to insertion of large number of elements to bit arrays. Moreover, there is no need to generate and insert queries of all possible lengths to a TSBF. As a query length analysis [19] on various search engines conducted in 2011 indicate, an average length of a query contains 3.08 terms and more than 92% queries contain only 5 terms or less. Moreover, there may be concept combinations that end user is simply not interested in as queries and the number of such concept combinations could be unnecessarily large. Therefore we propose that a peers dynamically build TSBF level 2 based on the queries it receives by adding only those queries that it can answer using its documents or through neighborhood. This will significantly lower the useless concept combinations in a TSBF level 2 thus controlling the false positive rate from going up. A peer also has the capability of aggressively building its TSBF level 2 by periodically gossiping with neighbors to exchange their query histories to discover new concept combinations that should be in its TSBF.

4.2. Routing Index

The routing index of a peer summarizes information of its neighborhood useful for intelligent query routing. One of the key requirements of the routing index of a peer is to represent all possible queries answerable by a neighbor along with the corresponding relevance strengths of the neighbor per each individual query in a compact manner. The routing index structure achieves this by summarizing TSBF's of neighbors at the peer. The routing index of a peer P consist of a set of *<key, value>* pairs where key is a neighbor N and value is a $TSBF_{P\rightarrow N}$, a summarized TSBF that represents the set of documents reachable through N and the set of queries answerable through the N based on the set of peers in the network N can reach. $TSBF_{P\rightarrow N}$ generally represents an aggregation of a set of $TSBF_{P'}$ structures each belonging to a peer P′ reachable through P.

4.2.1 Routing Index Updation

The drastic reduction in the overheads of routing table construction and maintenance in BSI is achieved through use of TSBFs. At startup, peers exchange their TSBFs with each other. Therefore the entry for neighbor N, $TSBF_{P\rightarrow N}$, at P's routing index is initially a replica of TSBFN at N representing the set of documents and queries in N's local storage. This is later updated by information carried by N to P from different query routes to represent documents and queries reachable through N. Particularly, presented with piggybacked information carried in a query path, an update at P aggregates $TSBF_{Pi}$ of each peer P_i visited by the query received through N, by taking the union equivalent of them subjecting bit counts to exponential decay depending on the hop distance of P_i to P. The exponential decay of counts in bits during forwarding of search messages, exponentially reduces the impact of a peer's TSBF on another peer's routing index with the distance between the two.

Equation 3 gives the exponential decay based union for combining TSBFs of set of peers N_j for a concept c. x here represents the level of the TSBF(i.e.$x \in \{1,2\}$) and α is the decay factor. We simply used the distance between peer N_j to P as α. Here the corresponding bits of respective bit arrays are aggregated using a function F. F is simply bitwise OR between bit i of corresponding bit arrays in all TSBFs in the query path for level 1 TSBFs and is sum for the same for level 2 TSBFs bit arrays. To avoid information becoming stale, periodic updates are triggered by peers updating Routing index for each neighbor based on the messages it received. Even though the updates are triggered periodically, the updates occur only if a peer sees a substantial difference in the values in the bits in bit arrays it is interested in.

$$TSBF_{x,expUnion}\,(c).bit_i = F_{j=1}^{k}\left(\frac{TSBF_{x.N_j}(c).bit_i}{\alpha}\right) \qquad (3)$$

The above design achieves our primary objective of efficiently maintaining probabilistic information about the content stored in the neighborhood. Algorithm 1 summarizes the routing index creation and updating procedure.

4.2.2 Routing Index Compression

To observe dynamically changing space restrictions, a peer can simply compress routing index at will by exploiting ontology IS-A structure. The main intuition here is that, due to IS-A relations, the ancestor concepts in an ontology hierarchy is fully qualified to represent a descendant concept. Therefore the $TSBF_{P\rightarrow N}$ of a neighbor N in P's routing index can be compressed even further by combining the bit arrays for two or more concepts sharing a IS-A relation by applying union equivalent of represented sets to corresponding bit arrays. for example, corresponding level bit arrays of B and D in ontology in Figure 1 can be combined to generate one bit array represented at D this way.

This union equivalent operation for a set of such TSBFs for a neighbor N at P representing a combining set of concepts sharing IS-A relations is given in Equation 4. The level x is 1 or 2 and C_a is the ancestor concept of all descendant concepts c_j being aggregated. Similar to Equation 3, F refers to bitwise or and summation of counts for level 1 and level 2 respectively.

$$TSBF_{x,union}\ (c_a).\,bit_i\ =\ F_{j=1}^{k}\big(TSBF_{x,P\rightarrow N}(c_j\,).\,bit_i \qquad (4)$$

When the need arises to compress a routing index, the priority first goes to combining bit arrays for different concepts at level 2 (i.e. $TSBF_{2,P}$) as compressing this does not result in minimal loss of information. The reason for this is, $TSBF_{2,P}$ represent exact information regarding frequency of concept combinations. The second priority is given to combining level 1 $TSBF_{1,P}$ bit arrays. Since this is used for estimation purposes, combining multiple bit arrays for different concept into one will degrade the estimation accuracy. Similarly, when choosing which bit arrays to combine in a given level, the higher probability is given for those bit arrays representing higher level concepts closer to the root as the information represented by the ontology generalizes towards ontology root.

Combining multiple bit arrays into one, however, results in slightly increased false positive rate and can be controlled by either not having excessively large number of items in bit arrays or by allowing bit arrays to grow as needed to maintain the desired false positive rate.

Algorithm 1 *Routing Index Construction and Maintenance*

```
Input:
  Msg = Message
  N = Neighbor sending Msg
  TSBFList = The list of TSBFs in Msg, list index+1 denote distance between P and peer to
             which TSBF at index belong
  P = Message receiver peer executing algorithm
  C = set of concepts in reference ontology

Output: ‘
  selected = the top k
     ∇ Create index entry for N for alive-ping message
1:
2:   if Msg.TYPE = ALIVE then
3:     TSBFN  ← TSBFList.get(0)
4:     for each Concept c ∈ C do
5:       for i = 0 to TSBFN(c).length - 1 do
6:         index:$TSBF_{1,P}$→N(c).$bit_i$ ← $TSBF_{1,N}$ (c).$bit_i$
7:         index:$TSBF_{2,P}$→N(c).biti ← $TSBF_{2,N}$ (c).$bit_i$
8:       end for
9:     end for
10:  end if
11:  ∇ Remove index entry for N for leave-ping message
12:  if Msg.TYPE = LEAVE then
13:    Index.remove(N)
14:  end if
15:  ∇ Update index entry of N for information received through a search related message
```

```
16: **if** Msg.Type = QUERY or HIT or MISS **then**
17:   ∇ *Determine concepts for which Routing index should be updated*
18:   Query q ← Msg.query
19:   lca(q) ← least common ancestor concept of q
20:   $C_{update}$ ← empty set representing concepts for which index.$TSBF_P$ (N) should be updated
21:   $C_{update}$.add(lca(q))
22:   $C_{update}$.addAll(q)
23:   ∇ *Update Routing index entry for N*
24:   **for each** Concept c $\in$ $C_{update}$ **do**
25:     ∇ *Calculate a new TSBF entry for N based on TSBFList*
26:     $TSBF_{1,N'}$(c) ← calculate using Equation 3
27:     $TSBF_{2,N'}$(c) ← calculate using Equation 3
28:     **if** $TSBF_{1,N'}$(c).$bit_i$ > index.$TSBF_{1,P\rightarrow N}$(c).$bit_i$ **then**
29:       Index.$TSBF_{1,P\rightarrow N'}$(c).$bit_i$ ← $TSBF_{1,N'}$(C).$bit_i$
30:     **end if**
31:     **if** $TSBF_{2,N'}$ (c).$bit_i$ > index.$TSBF_{2,P\rightarrow N}$(c).$bit_i$ **then**
32:       Index.$TSBF_{2,P\rightarrow N}$(c).$bit_i$ ←$TSBF_{2,path}$(C).$bit_i$
33:     **end if**
34:   **end for**
35: **end if**
```

4.3. Query Routing

The query routing algorithm is summarized in Algorithm 2. The objective of search in BSI is to retrieve many relevant documents as possible for a given query. The search messages in BSI are TTL bounded and therefore a query can travel maximum TTL hops before it is discarded. Every query receiver peer performs a local search and append results of retrieved documents to the query and then forwards the query to the most promising neighbor if the TTL has not expired. Upon TTL expiration, the peer returns a response message to the query originator along with the retrieved results. The neighbor selection process at a peer P for a query q is performed as follows: Upon receipt of q, P checks $TSBF_{P\rightarrow N}$ associated with each of its neighbor N. The lookup returns the frequency of documents (exceeding a relevance threshold) in each neighbor N containing query as a concept combination. This denotes the relative probability of success of N for the query compared to its neighbors. To calculate this TSBF, P first lookup in level 2, $TSBF_{P\rightarrow N}$ *(c)*, where c is the least common ancestor of concepts in query. If query exist as an element, $TSBF_{P\rightarrow N}$ returns the associated document frequency. The fact that $TSBF_{2,P\rightarrow N}$ *(c)* does not contain q as a concept combination does not necessarily mean that combination is not reachable through N. It may be a situation arising from P not exploiting all reachable peers in its TTL hop radius or P has not encountered q in its query history. In such a case P calculates an estimate of the reachable documents for a multi-concept queries based on $TSBF_{2,P\rightarrow N}$ from its routing index. This estimation procedure is described below.

1) Estimating set intersection based cardinality from Bloom filters: We need a reliable mechanism to find the number of documents reachable through a neighbor for a given query solely based on the set of Bloom filters each representing documents reachable through the neighbor for each query concept. Research community has proposed many works to estimate the cardinality(i.e. number of elements) of an original set solely based on its Bloom filter bit array [14], [16], [17]. For our work we used the work presented by authors of [16].

Given a set S and its Bloom Filter BFS with m bits out of which t are true bits, and k hash functions [16] defines the cardinality of a set is as follows:

$$|S| = \frac{ln\left(1-\frac{t}{m}\right)}{k \times ln\left(1-\frac{1}{m}\right)} \tag{5}$$

Here, the m denotes the length of the Bloom filter and t the number of true bits in the Bloom filter. |S| denotes the cardinality of the set represented by the Bloom filter bit array.

For cardinality estimation of union of multiple sets $S_\cup = S_1 \cup S_2 \cup \ldots \cup S_n$ (needed for OR query routing) authors simply propose to take the bitwise OR of the representative Bloom filter bit arrays and apply equation 5 to the resulting Bloom filter. Estimating cardinality of intersection of sets based on bloom filters is not so straightforward. We used the set inclusion-exclusion principle in combinatorics to derive the cardinality of intersection of sets based on the cardinalities of the unions of sets. We employed the equation 5 to calculate the intersection of sets $S_\cap = S_1 \cap S_2 \cap \ldots \cap S_n$:

$$\left|\bigcap_{i=1}^{n} S_i\right| = \sum_{k=1}^{n} (-1)^{k+1} \left(\sum_{1 \le i < k \le n}^{v} |S_i \cup \ldots S_k| \right) \tag{6}$$

Each set S_i in our work represent the set of documents reachable for each concept c_i for a given query through a given peer P. The equation 5 only requires cardinalities of individual sets and unions of sets to compute the cardinality of intersection. Therefore upon a receipt of a query a peer can simply use above mentioned Bloom filter based cardinality estimation techniques to calculate each term in right hand side of Equation 6 thus computing the cardinality of intersection for a neighbor for a given query.

Algorithm 2 *Query Routing*

```
Input:
  Q = Query Message
  P = Message receiver peer
  index = The routing index of P
  Candidates(Q) = Neighborhood(P) - Q.VisitedPeers
  k = the walker count
Output: ‘
  R = Retrieved documents
  Selected(Q) = the selected peers for query forwarding

1:   Selected(Q) $\leftarrow \phi$
2:   Q.TTL $\leftarrow$ Q.TTL - 1
3:   R $\leftarrow$ LocalSearch(o)
4:   $\nabla$ Found target Object
5:   if Q.TTL $\le 0$ then
6:     if R $\neq \phi$ then
7:       Return a HIT with R and terminate search
8:     Else
9:       Return a MISS with R and terminate search
10:    end if
```

```
11: Else
12:   $C_{lca} \leftarrow$ least common ancestor concept of Q
13:   for each Neighbor N $\in$ Candidates(Q) do
14:     $\nabla$ Look up score in index.$TSBF_{2,P \rightarrow N}$
15:     if index.$TSBF_{2,P} \rightarrow N(C_{lca})$ contains Q then
16:       $Score_N \leftarrow$ minimum count of all bits hashed by Q in index.$TSBF_{2,P \rightarrow N}(C_{lca})$
17:     else $\nabla$ Estimate score from index.$TSBF_{1,P \rightarrow N}$
18:       $QueryBFList_N \leftarrow \phi$
19:       for each Concept c $\in$ Q do
20:         $QueryBFList_N$.add(index. $TSBF_{1,P \rightarrow N}(c)$)
21:       end for
22:       $Score_N \leftarrow$ calculate using Equation 6 with $QueryBFList_N$ as input
23:     end if
24:     Sort Candidates(Q) based on scores
25:     Selected(Q) $\leftarrow$ Neighbor in Candidates(Q) with highest score
26:   end for
27:   Return Selected(Q) and R
28: end if
```

5. EXPERIMENTS

In this section we present a simulation-based evaluation of BSI and compare its performance with other mechanisms for query routing in unstructured peer to peer systems.

5.1. Simulation Methodology

The parameters for the simulation TSBF used in BSI are listed in Table 1. The simulations use a default 1024 peer unstructured network, unless noted otherwise. To simulate the dynamic behavior of the network under peer churn, we inserted online nodes to the network while removing active nodes at varying frequencies. During a single simulation run, the network size was maintained at roughly the original starting size by ensuring the number of nodes that join the network dynamically was the same as that leaving the network. On an average, 80 nodes each are added and removed from the network during each simulation run.

We mainly compare our work against OSQR [7] and Random Walk [8]. OSQR is a concept based indexing mechanism which tries to improve the performance for multi-concept queries with a concept based routing index. The OSQR routing index of a peer maintains the total reachable documents per neighbor per concept in a reference ontology as the relevance strengths of the neighbor for each ontology concept. The authors propose taking the minimum of the number of documents reachable for each query concepts for a given neighbor as its relevance strength to the query. For the simulation, the maximum number of entries in OSQR index was set to 128 and size of *maxVector* (a data structure used by peers for global knowledge acquisition) was set to 10. Random walk on the other hand is a popular blind search mechanism where a querying peer deploys a search walkers by sending query to a randomly selected neighbor. The query format as well as local search remains the same as our proposed BSI algorithm. We experimented with two versions of BSI based on the design of the TSBF data structure:

1) BSI(TSBF-level1+level2): This is the original BSI algorithm where the TSBF structure contains two levels: level1 contains a list of bit arrays each representing the set of documents

accessible per ontology concept and level2 contains a list of bit arrays each representing a set of queries and the associated document frequencies per ontology concept of a peer.
2) BSI(TSBF-level1): This represents a reduced level BSI algorithm where TSBFs solely consists of level1 only. Thus the peers rely on the reachable documents estimation process in query routing.

Table 1. Simulation Parameters.

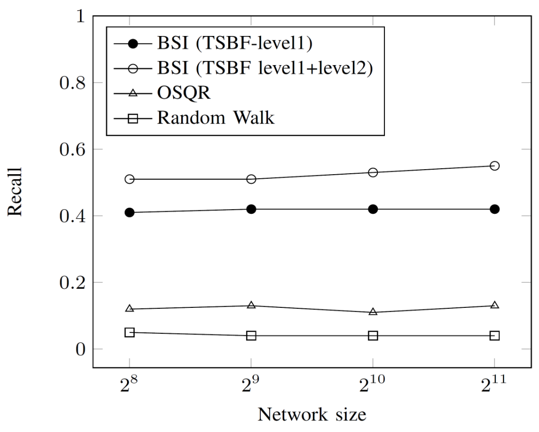
Parameter	**Range and default value**
Network size	2^8-2^{11} Default:2^{10}
Peer degree distribution	power law
Total distributed documents	5000
Average number of concepts per document	20
Document distribution among peers	zipf (α=1.0)
Query distribution among peers	zipf (α=1.2)
Mean peer documents	100
Reference Ontology Concepts	128
TTL	1-11 Default:7
Relevance threshold	0.7
TSBF filter length	250bits
No. of hash functions	7

5.2. Results

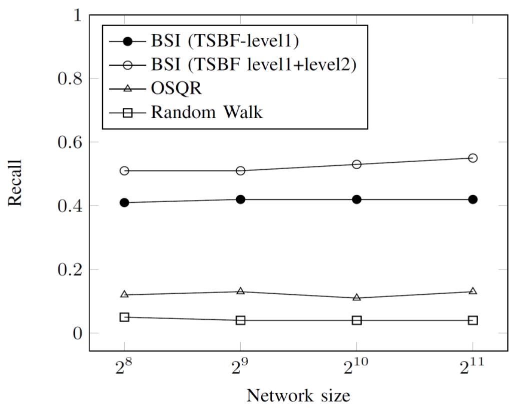
The primary performance results can be summarized as follows. For a low routing overhead, the query recall improves by up to five orders of magnitude over OSQR proving the efficiency of the algorithm in locating as many relevant documents as possible in a low search cost (Section 5.2.1). Simulations over different query lengths and different filter sizes demonstrate the scalability of BSI for multi-term query routing in sections 5.2.2 and 5.2.3 respectively. On average we observed up to four fold performance improvement in terms of recall over OSQR for various query lengths and different filter sizes. Finally, a study of the impact of search cost on overall search efficiency in section 5.2.4 demonstrates that the performance improvement in recall is achieved at an attractive two-fold improvement in search traffic over OSQR.

5.2.1 Recall

Recall is a standard performance measurement in information retrieval that denotes the fraction of relevant documents found by an average query compared to the entire set of relevant documents distributed in the network. While many query routing mechanisms can achieve high recalls at sufficiently high hop limits only superior query routing algorithms can achieve high recalls with low hop limits. Varying the hop limit allows us to demonstrate the effectiveness of our algorithm in finding larger number of documents at low TTL values thus with lower search cost. Figure 4(a) plots the average recall per query for BSI, OSQR and Random Walk. We observed that original BSI (i.e. (TSBF-level1+level2)) algorithm achieved an average of 383.71% increase in recall over OSQR while BSI(TSBF-level1) gained a 322.60% increase in recall over OSQR. The performance improvement of BSI(TSBF-level1+level2) over Random walk was observed to be an outstanding 1238.81%. The increase in recall is more evident in higher TTL values. From the figure we can also infer that the original BSI with both levels present in the TSBF structure performs better than BSI with the TSBF with only level1. An average increase of 45.16% improvement was observed in BSI (TSBF level1+level2) over BSI (TSBF-level1). This is expected, as having both levels in a TSBF obviously provide more information regarding query relevance strength of a peer to make a intelligent query routing decision compared to a TSBF level 1 which only provide an estimate of the same. As Figure 4(b) suggests, both versions of BSI clearly outperforms OSQR and Random Walk in terms of recall regardless of the network size thus proving the scalability of our algorithm.



(a) Recall vs. TTL



(b) Recall vs. Network Size

Figure 4. Recall for Single Concept Query (Filter Size = 250bits)

5.2.2 Effect of Query Length

Effect of Query Length: One of the key design objectives of BSI is to achieve high recalls for queries with more than single concept. Figure 5 plots the effect of query length on recall. We observed a significant improvement in recall in BSI over OSQR for both BSI versions: BSI(TSBF-level1+level2), and BSI(TSBF-level1). On average for all query sizes we observed that recall improves four orders of magnitude (315.79%) and by three orders of magnitude (223.88%) compared to OSQR in BSI(TSBF-level1+level2) and BSI(TSBF-level1) respectively. Recall improves ten orders of magnitude (903.62%) and by three orders of magnitude (682.01%) compared to Random Walk in BSI(TSBF-level1+level2) and BSI(TSBF-level1) respectively. The superiority of BSI was more evident for queries with higher number of concepts. The reason for this behavior can be explained using the peer knowledge summarization and the neighbor selection mechanisms of all three algorithms. Random walk does not use any intelligence in neighbor selection mechanism. Peers randomly select one if their neighbors to forward queries. BSI and OSQR estimate the number of documents reachable thorough a peer for a query as its relevance score in neighbor selection process. An OSQR peer only maintains the number of reachable documents per ontology concept for each neighbor in its routing index and a peer estimates the number of documents reachable from a given neighbor by taking the minimum of the number of documents reachable for each queried concept for that neighbor. Taking the minimum as the relevance score, however, is an optimistic estimate. This actually represents an upper bound of the number of actual relevant documents reachable through a neighbor.

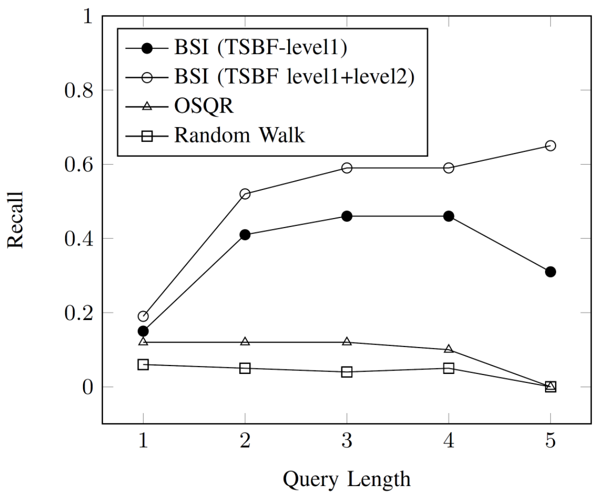


Figure 5. Recall vs. Query Length (TTL=7, Filter Size=250bits).

To take a more accurate decision, the decision taking peer must ideally have the set of documents reachable through the neighbor for each queried concept and should perform a intersection of all sets of documents relevant to the query to obtain the set of documents actually reachable through that neighbor for the query. The size of this resulting set will accurately represent the number of documents reachable through the neighbor for that query. However, maintaining such detailed information is too costly in P2P networks. BSI compensates this information loss by maintaining bloom filters representing the queries (along with number of documents reachable as associated bit counts) and documents reachable through a neighbor, thereby providing a more compact and high quality information summarization leading to more accurate estimations of number of documents reachable for queries.

5.2.3 Effect of Filter Size

Having established the scalability of our general approach, we now turn our attention to the effect of Bloom filter size in recall and storage cost.

Using TSBFs for local knowledge maintenance results in substantial increase in recall at low storage cost. As shown in Figure 6 recall substantially increases with filter size in all BSI versions. This is the expected behavior as larger filter sizes can keep large volume of information at a tolerable false positive rate resulting in better query routing decision leading to high recall rates. On average we observed a 300.47% and 216.37% average improvement in recall over OSQR and a 1038.60% and 799.50% average improvement in recall over Random Walk for BSI (TSBF level1+level2) and BSI (TSBF-level1) respectively.

Table. 2 shows the associated storage costs for various filter sizes. This storage cost accounts for a peer's TSBF and its routing index. Storage costs are shown for original BSI(TSBF with level 1 and level 2 present) and for two modified BSI versions where a TSBF contain only level1 or level 2 respectively. While maintaining a large filter size allows us to decrease the false positive rate we observed that it substantially increases the TSBF size. However, while storing a TSBF in a peer's routing index per neighbor can contribute to increase in index sizes, this storage requirement is not a burden to an average internet computing device. Moreover, as long as the false positive rate of the TSBF is at an acceptable level for the application, there is no need to unnecessarily increase the filter size. We found through experimentation that the optimal filter size for our trace was approximately 250 bits. However, those peers that have storage limitations and cannot afford to reserve a filter size below this size have the option of either compressing their routing indices as illustrated in Section 4.2.2 or to eliminate one of the levels of TSBFs stored in routing indices to observe their storage restrictions.

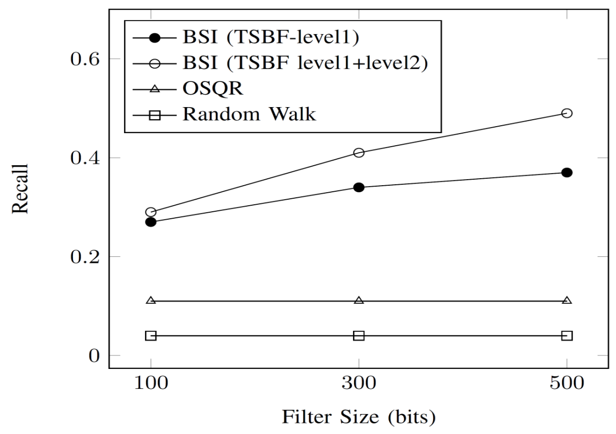


Figure 6. Recall vs. Filter Size (TTL=7, Average Query Length=2.5)

Table 2. Storage Overhead.

Algorithm	**Filter size (bits)**		
	100	**300**	**500**
OSQR	10752.01	10752.01	10752.01
BSI (TSBF level 1+level 2)	4875.00	14625.00	24375.00
BSI (TSBF-level 1)	4800.00	14400.00	24000.00
BSI (TSBF-level 2)	56.25	168.75	281.25

5.2.4 Search Overhead

Using TSBFs for knowledge transfer and acquisition process results in significant lowering of data transmission. OSQR consumes a significant amount of bandwidth in data transfer. OSQR on average consumed 13.94KB of bandwidth per query in data transfer. This was in contrast to the 4.13KB and 3.88KB consumed by a query in BSI (TSBF level 1+level 2)and BSI (TSBF-level 1) respectively. This corresponded to a 70.35%, and 95.46% reduction over OSQR in traffic cost for BSI (TSBF level 1+level 2)and BSI (TSBF-level 1) respectively. Random Walk search message consumed only 1.02KB bandwidth as a message does not carry additional information for knowledge dissemination. The high search cost of OSQR is attributed to the fact that it transmits routing index information about each taxonomy concept for every visited neighbor as feedback. Queries in BSI on the other hand transfers much more space efficient TSBFs relevant only to the queried taxonomy concepts thus significantly reducing the bandwidth consumption.

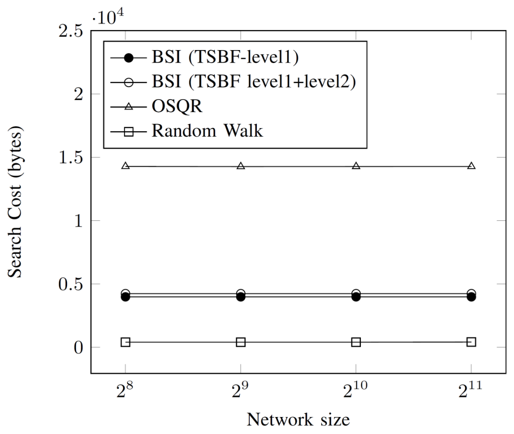


Figure 7. Search Cost (TTL=7, Filter Size=250bits, Average Query Length=2.5)

To get an accurate picture of overall search efficiency of a search algorithm we need a combined measure of both recall and search cost. Therefore we define the search efficiency to be the ratio of recall with search cost. In Figure 8 we plot the search efficiency of BSI, OSQR and Random Walk against network size. From the results it is clear that efficiency of BSI (both versions) is much higher compared to OSQR and Random Walk, and BSI is capable of producing high recall at much lower message cost.

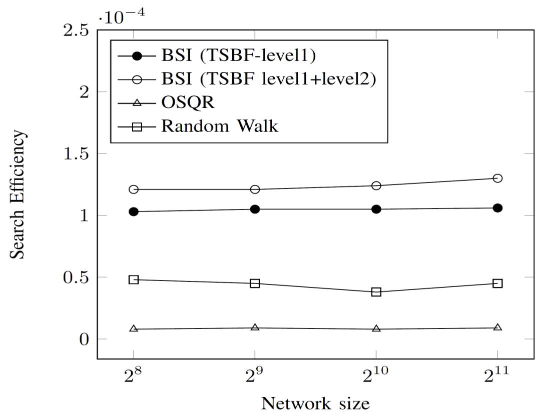


Figure 8. Search Efficiency (TTL=7, Filter Size=250bits, Average Query Length=2.5)

6. CONCLUSION AND FUTURE WORK

Routing in unstructured P2P networks is a challenging problem. In this work, we have explored the approach of compact representation of large semantic query based indices through the use of Bloom filters. Our Bloom filter based routing index quantifies the strength of a peer's neighborhood for individual queries and then uses this information for forwarding queries. The simple yet powerful design of BSI makes it easy to implement. Simulation based comparison with state-of-the art query routing mechanisms, under a wide variety of scenarios establish the performance advantages of BSI. The BSI search framework presents interesting possibilities of implementing high level semantics of trust, reliability, etc. using routing and forwarding policies.

REFERENCES

[1] Napster website. http://www.napster.com/, 2006.

[2] Gnutella website. http://www.gnutella.com/, January 2007.

[3] Madhan Arumugam, Amit Sheth, and I. Budak Arpinar. "Towards peer-to-peer semantic web: A distributed environment for sharing semantic knowledge on the web". Technical report, 2001.

[4] Michael Bendersky and W. Bruce Croft, "Discovering key concepts in verbose queries" In Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, ACM SIGIR '08, pages 491–498, New York, NY, USA.

[5] Michael W. Berry, Zlatko Drmac, and Elizabeth R. Jessup. "Matrices, vector spaces, and information retrieval" In SIAM Rev., 41(2):335–362.

[6] Arturo Crespo and Hector Garcia-Molina. "Semantic overlay networks for p2p systems" In Proceedings of the Third international conference on Agents and Peer-to-Peer Computing, AP2PC'04, pages 1–13, 2004.

[7] Rasanjalee Dissanayaka, S.B. Navathe, and Sushil K. Prasad. "OSQR: A framework for ontology-based semantic query routing in unstructured p2p networks" In Proceedings of international IEEE HiPC conference on High Performance Computing, HiPC '12, 2012.

[8] Christos Gkantsidis, Milena Mihail, and Amin Saberi. "Random walks in peer-to-peer networks: algorithms and evaluation" Perform. Eval., 63(3):241–263, 2006.

[9] Hai Jin and Hanhua Chen. "Semrex: Efficient search in a semantic overlay for literature retrieval" Future Gener. Comput. Syst., 24(6):475–488, 2008.

[10] Vana Kalogeraki, Dimitrios Gunopulos, and D. Zeinalipour-Yazti. "A local search mechanism for peer-to-peer networks" In Proceedings of the eleventh international conference on Information and knowledge management, ACM CIKM '02, pages 300–307, 2002.

[11] Juan Li and Son Vuong. "Ontsum: A semantic query routing scheme in p2p networks based on concise ontology indexing" In Proceedings of the 21st International Conference on Advanced Networking and Applications, IEEE AINA '07, pages 94–101, 2007.

[12] Qin Lv, Pei Cao, Edith Cohen, Kai Li and Scott Shenker. "Search and replication in unstructured peer-to-peer networks" In Proceedings of the 16th international conference on Supercomputing, ICS '02, ACM, pages 84–95, New York, NY, USA, 2002.

[13] H. Mashayekhi, J. Habibi, and H. Rostami. "Efficient semantic based search in unstructured peer-to-peer networks" In Modeling Simulation, 2008. AICMS 08. Second Asia International Conference on, pages 71–76, 2008.

[14] Loizos Michael, Wolfgang Nejdl, Odysseas Papapetrou, and Wolf Siberski. "Improving distributed join efficiency with extended bloom filter operations" In AINA IEEE, pages 187–194, 2007.

[15] Wolfgang Nejdl, Boris Wolf, Changtao Qu, Stefan Decker, Michael Sintek, Ambj¨orn Naeve, Mikael Nilsson, Matthias Palm´er, and Tore Risch. "Edutella: a p2p networking infrastructure based on rdf" In Proceedings of the 11th international conference on World Wide Web, ACM '02, pages 604–615, New York, NY, USA, 2002.

[16] Odysseas Papapetrou, Wolf Siberski, and Wolfgang Nejdl. "Cardinality estimation and dynamic length adaptation for bloom filters" Distrib. Parallel Databases, 28(2-3):119–156, December 2010.

[17] Sukriti Ramesh, Odysseas Papapetrou, and Wolf Siberski. "Optimizing distributed joins with bloom filters" In Proceedings of the 5th International Conference on Distributed Computing and Internet Technology, ICDCIT '08, Springer-Verlag, pages 145–156, Berlin, Heidelberg, 2009.

[18] Habib Rostami, Jafar Habibi, Hassan Abolhassani, Mojtaba Amirkhani, and Ali Rahnama. "An ontology based local index in p2p networks" In Proceedings of the Second International Conference on Semantics, Knowledge, and Grid, SKG '06, IEEE, pages 11–, Washington, DC, USA, 2006.

[19] Mona Taghavi, Ahmed Patel, Nikita Schmidt, Christopher Wills, and Yiqi Tew. "An analysis of web proxy logs with query distribution pattern approach for search engines" Computer Standards & Interfaces, 34(1):162 – 170, 2012.

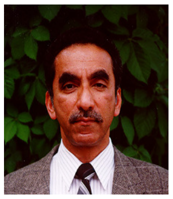
[20] Chunqiang Tang, Zhichen Xu, and Mallik Mahalingam. "psearch: information retrieval in structured overlays" SIGCOMM Comput. Commun. Rev., 33(1):89–94, January 2003.

[21] Beverly Yang and Hector Garcia-molina. "Efficient search in peer-to-peer networks" In Proceedings of the 22nd International Conference on Distributed Computing Systems, 2002.

AUTHORS

Rasanjalee Dissanayaka received her Ph.D. in Computer Science from Georgia State University in 2014. She is currently serving as an Assistant Professor of Computer Science at College of St. Benedict & St. Johns University since August 2014. Her main research interests are in the areas of distributed algorithms and semantic computing . Rasanjalee's publications include papers in international journals and conference proceedings.

Sushil K. Prasad (BTech'85 IIT Kharagpur, MS'86 Washington State, Pullman; PhD'90 Central Florida, Orlando - all in Computer Science/Engineering) is a Professor of Computer Science at Georgia State University (GSU) and Director of Distributed and Mobile Systems (DiMoS) Lab. He has carried out theoretical as well as experimental research in parallel and distributed computing, resulting in 120+ refereed publications, several patent applications, and about $3M in external research funds as principal investigator and over $6M overall (NSF/NIH/GRA/Industry).



Shamkant Navathe is currently a professor at the College of Computing, Georgia Institute of Technology, Atlanta. He is known for his work on database modeling, database conversion, database design, distributed database fragmentation, allocation, and redesign, and database integration. Current projects include biological genome databases, data mining, modeling of database security, and knowledge base design. He has worked with IBM and Siemens in their research divisions and has been a consultant to various companies, including Honeywell, Nixdorf, CCA, ADR, Digital, MCC, Harris, Equifax and Hewlett Packard corporations.

Vikram Goyal is a faculty in IIIT Delhi. He received his PhD in Computer Science and Engineering from the Department of Computer Science and Engineering at IIT Delhi in 2009. Before pursuing PhD, he completed his M.Tech in Information Systems from the Department of Computer Science and Engineering at NSIT Delhi in 2003. His primary area of research is Spatial-Textual Data Management and Data Privacy. He has been given a DST Young Scientist Award for his research on data privacy in Location-based Services.