Spring 5-10-2014

Ontology-based Search Algorithms over Large-Scale Unstructured Peer-to-Peer Networks

Rasanjalee Dissanayaka Mudiyanselage

ONTOLOGY-BASED SEARCH ALGORITHMS OVER LARGE-SCALE UNSTRUCTURED PEER-TO-PEER NETWORKS

by

RASANJALEE HIMALI DISSANAYAKA MUDIYANSELAGE

Under the Direction of Sushil Prasad and Shamkant Navathe

ABSTRACT

Peer-to-Peer(P2P) systems have emerged as a promising paradigm to structure large scale distributed systems. They provide a robust, scalable and decentralized way to share and publish data.The unstructured P2P systems have gained much popularity in recent years for their wide applicability and simplicity. However efficient resource discovery remains a fundamental challenge for unstructured P2P networks due to the lack of a network structure. To effectively harness the power of unstructured P2P systems, the challenges in distributed

knowledge management and information search need to be overcome. Current attempts to solve the problems pertaining to knowledge management and search have focused on simple term based routing indices and keyword search queries. Many P2P resource discovery applications will require more complex query functionality, as users will publish semantically rich data and need efficiently content location algorithms that find target content at moderate cost. Therefore, effective knowledge and data management techniques and search tools for information retrieval are imperative and lasting.

In my dissertation, I present a suite of protocols that assist in efficient content location and knowledge management in unstructured Peer-to-Peer overlays. The basis of these schemes is their ability to learn from past peer interactions and increasing their performance with time.My work aims to provide effective and bandwidth-efficient searching and data sharing in unstructured P2P environments. A suite of algorithms which provide peers in unstructured P2P overlays with the state necessary in order to efficiently locate, disseminate and replicate objects is presented. Also, Existing approaches to federated search are adapted and new methods are developed for semantic knowledge representation, resource selection, and knowledge evolution for efficient search in dynamic and distributed P2P network environments. Furthermore,autonomous and decentralized algorithms that reorganizes an unstructured network topology into a one with desired search-enhancing properties are proposed in a network evolution model to facilitate effective and efficient semantic search in dynamic environments.

INDEX WORDS: Peer-to-Peer networks, Query routing, Semantic search, Unstructured overlay, Indexing, Semantic clustering, Replication

ONTOLOGY-BASED SEARCH ALGORITHMS OVER LARGE-SCALE UNSTRUCTURED PEER-TO-PEER NETWORKS

by

RASANJALEE HIMALI DISSANAYAKA MUDIYANSELAGE

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2014

ONTOLOGY-BASED SEARCH ALGORITHMS OVER LARGE-SCALE UNSTRUCTURED PEER-TO-PEER NETWORKS

by

RASANJALEE HIMALI DISSANAYAKA MUDIYANSELAGE

Committee Chair:	Sushil K Prasad
Committee Co-Chair:	Shamkant Navathe
Committee:	Rajshekhar Sunderraman
	WenZhan Song

Electronic Version Approved:

Office of Graduate Studies
College of Arts and Sciences
Georgia State University
May 2014

DEDICATION

This dissertation is dedicated to my loving and supportive husband, Jayampathi Sampath Rajapaksage, who has been proud and supportive of my work and who has shared the many uncertainties, challenges and sacrifices for completing this dissertation. I am truly thankful for having you in my life. This dissertation is also dedicated to our sweet, and kind-hearted little son, Saven Thisev Rajapaksage, who has grown into a wonderful almost two year old in spite of his mother spending so much time away from him working on this dissertation. Finally, this dissertation is also dedicated to my always encouraging, ever faithful parents, Mallika Kahaduwa and D M Wimalasiri. Thank you for all the unconditional love, guidance, and support that you have always given me.

ACKNOWLEDGEMENTS

This thesis is the result of research carried out over a period of two years. During this period, many people supported my work and helped me to bring it to a successful conclusion, and here I would like to express my gratitude.

First, I would like to express my gratitude to my advisors, Dr. Sushil K. Prasad, and Dr. Shamkant Navathe for their support and invaluable guidance throughout my study. Their knowledge, perceptiveness, and innovative ideas have guided me throughout my graduate study.

I also present my words of gratitude to the other members of my thesis committee, Dr. Rajshekhar Sunderraman, and Dr. WenZhan Song, for their advice and their valuable time spent in reviewing the material. I especially want to express my sincere gratitude to Dr. Sunderraman for all his help, advice and support during my graduate studies. I deeply appreciate his valuable advice and support on both professional and personal matters.

I also would like to extend my appreciation to my colleagues in DiMoS research group for all the support and ideas and to everyone who offered me academic advice and moral support throughout my graduate studies.

Finally, I would like to express my gratitude to my family, especially my husband and my parents, for their unconditional support and outstanding belief in my success. Without their support, this research project would not have been possible. Their continuous support played an essential role in helping to evolve my ideas and improve the quality of this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

- P2P - Peer-to-Peer
- BF - Bloom Filter
- RL - Reinforcement Learning

PART 1

INTRODUCTION

1.1 Peer-to-Peer Networks

P2P networks have emerged as a result of the recent advances of storage and networking technologies to provide data sharing and circulation through internet for large number of users scattered all over the world. They can broadly be classified into structured and unstructured networks. The links between peers in an unstructured network is arbitrary and there is no relationship between the data and where peer resides. The unstructured nature of the network, makes searching a challenging problem. Many popular P2P networks such as Gnutella [1] and KaZaA [2] are unstructured P2P systems. Peers in structured P2P networks on the other hand have more structured pattern of overlay links to ensure guaranteed search lookups. Structured P2P networks however suffer from high maintenance overhead in maintaining overlay links in frequent peer join and leaves inherent to P2P systems.

The popularity of P2P networks for file sharing applications comes from numerous advantages it offers including low maintenance overhead, high scalability and reliability, synergistic performance, increased autonomy, privacy and dynamism. According to very conservative estimates [3], there exist more than 10×10^9 MHz of CPU power and 10,000 TB of storage not utilized at the edges of the Internet. According to [4] , bandwidth consumption attributed to popular file-sharing applications amounts to a considerable fraction (up to 60%) of the total Internet traffic. This indicates that there is a vast amount of unused potential over the Internet and current resource-sharing applications are responsible for huge amounts of data transmissions over the network. P2P technology can play a key role in our efforts to tackle both issues.

P2P systems are popular for large scale information retrieval and search in huge volumes of data dynamically distributed among many peers. While the data distributed in P2P

networks are largely unstructured, with the advent of semantic web, more and more users incorporate semantic meta-data with their resources. Therefore, effective search tools and techniques for semantic information retrieval are imperative and lasting.

1.2 Motivation

While much research has been devoted to P2P search in last several years [5–9], it is our belief that some critical areas of searching in purely decentralized information retrieval systems have not been thoroughly studied. This includes knowledge representation, heterogeneous data management, resource replication and decision making in presence of a partial view of entire system to name a few. With proper attention to these unaddressed aspects, P2P systems have great potential which could result in significant technological breakthroughs.

Here we summarize the main challenges and limitations unaddressed by the current body of work:

- *Knowledge Representation* : Users today comprehensively generate, manipulate and exploit shared content on the web by annotating semantic meta-data on web documents and performing reasoning on such data sources . This phenomena has to be adequately reflected in the knowledge representational models used in P2P networks . While we think in general the assumptions of knowledge representation in the Semantic Web are a good starting point, the knowledge representation in dynamic and distributed environments such as P2P networks generates special requirements such as ability gracefully handle peer dynamics and the need to capture the richness of peers in their semantic entities of interest. Quantifying the richness of semantic entities such as concepts, properties, and relations has not been addressed in the context of ontologies, although it is a crucial part of success in P2P search protocols. We identified three core questions which we try to tackle in the knowledge representation approach presented in this dissertation:

1. How can we support the tailoring of ontologies towards different needs in dynamic and distributed P2P applications such as frequent topology and content change and storage and bandwidth capacity limitations?
2. How can we represent uncertainty of knowledge inherent to P2P networks using current semantic knowledge representation models?
3. How can we cope with the heterogeneity of knowledge models and ontologies with potentially different interpretation schemes?

- *Complex Query Routing* : The more information we have on the Internet the more difficult it is to find terms of specific contexts. Simply searching for keywords is not enough anymore due to incorrect and incomplete keyword declarations and ambiguous words that exist in our natural languages. Users today should be able to pose more complex queries that can exploit the full power of semantic knowledge representation models. Even though complex semantic query languages such a RDQL and SPARQL exist today, end users are far from experienced in using such complex ontology base query languages. This mandates developing more simple and end user friendly query languages. There is also an inadequacy in simple yet powerful semantic knowledge representation models that can effectively augment probabilistic representation to current semantic technologies that allow representing the inherent uncertainty in knowledge in P2P systems. Little to no research done in integrating probability theory with current semantic web technologies. The efficiency of P2P search systems largely depends on the effectiveness of the knowledge representation model and quality of the reasoning mechanism used for query routing. Information retrieval mechanisms in large scale dynamic distributed systems like P2P should account for the fact that they operate based on incomplete knowledge acquired by peers in interested domain(s). Therefore, knowledge representation models should be capable of representing prior domain knowledge a peer acquires over time and the reasoning mechanism used for peer selection in query routing should draw probabilistic conclusions based on this knowledge.

Moreover, while many P2P query routing algorithms proposed so far exhibit a lack of perform in routing multi-term queries. As query logs of popular web search engines suggest, majority of queries issued by users contain more than single term. The naive solution of maintaining term-set based indices to achieve the purpose will simply result in exponential growth of indices. Therefore we need space efficient index structures that can hold large volumes of possible queries(i.e term sets) along with associated relevance strengths. The research questions that needs to be answered here are two-fold:

1. How can we exploit this potentially probabilistic knowledge in peers to derive new knowledge and probabilistic conclusions leading to effective query routing using semantic technologies?
2. How can the semantic knowledge models represent the knowledge necessary for routing multi-term queries while not resulting in exponential growth of routing indices?

- *Rare Content Search* : Unstructured P2P networks have poor search efficiency,especially for rare objects. Most of the existing search algorithms are very effective for locating popular objects but are far less effective in searching rare objects. As shown in [10], as much as 18% of all queries return no response even when results are available in the widely used Gnutella network. This calls for efficient rare object search with low communication and storage overhead.

1.3 Contribution

The present dissertation makes the following contributions:

- A survey of state of the art information retrieval approaches and knowledge representation approaches which are used in P2P networks.
- Design and development of a mechanism to measure the semantic richness of peers in considering multiple factors such as information content in shared documents, peer

connections in overlay structure, etc.

- A set of search protocols for completely distributed unstructured networks that combines the powers of both P2P systems and ontologies to improve search efficiency and interoperability in the network.

- A novel topology adaptation model that reorganize the network overlay based on their semantic interests expressed using a reference ontology.

- A novel content replication mechanism that proactively replicates rare objects in the network overlay for high search efficiency.

- Empirical evaluation of the developed algorithms using simulations.

1.4 Structure of the Thesis

The rest of this dissertation is structured as follows.

In Chapter 2 we present our system model and survey the related work, from P2P systems literature.

Chapter 3 introduces a probabilistic indexing framework that utilize peers past experience to guide future queries in a successful manner. The index is built upon the core concept of emergence of successful query paths in the P2P overlay as peers discover, (re)use query paths queries travel that find target object successfully developing a query traffic system. The indexing scheme is augmented with a rare object replication mechanism allowing us to handle rare content location gracefully while leveraging parts of existing structures. In Chapter 5 we present a novel ontology based indexing scheme to summarize and represent peers' semantic interests in a quantified manner. We also put forward a query routing algorithm which exploits the index in routing semantic queries in P2P environments. This semantic framework also provides a dynamic knowledge evolution mechanism as well as a low-overhead global knowledge acquisition mechanism that aims to improve the query precision over time.

In Chapter 4 we present a topology adaptation algorithm where peers are physically grouped based on their semantic interests. This semantic clustering framework utilize the concepts and the structural relations of a reference ontology to define peers' semantic interests as well as to reorganize the network so that P2P overlay clusters and connections mimic the concepts and the relations in the reference ontology. Query routing utilize the structural relations in ontology to route query to provide guaranteed look-ups in a steady-state network.

Web query logs shows that more than 60% of queries contain two or more terms. Majority of semantic search approaches, however, are devised for single concept queries. While maintaining all term combinations is simply impractical due to exponential growth of index size, simple cosine similarity measure or set intersection do not exploit the information offered by reference ontology to draw strong enough probabilistic conclusions. In Chapter **??** we present a small size multi-concept index that can efficiently route multi-concept queries in the semantic overlay. The proposed index structure compresses high dimensional semantic space into single dimension to maintain small index size while not compromising the quality of the index.

Chapter 7 summarizes the results obtained in this dissertation and shows directions for future work.

PART 2

RELATED WORK

P2P networks have emerged as a result of the recent advances of storage and networking technologies to provide data sharing and circulation through internet for large number of users scattered all over the world. The popularity of P2P networks for file sharing applications comes from numerous advantages it offers including low maintenance overhead, high scalability and reliability, synergistic performance, increased autonomy, privacy and dynamism.

P2P networks can broadly be classified into structured and unstructured networks based on the control over data location and network topology. Structured P2P networks have well defined neighbor links and thus provide guaranteed and bounded-time lookups. Examples of such systems include Chord [11], Pastry [12], and CAN [13]. A key disadvantage in structured P2P networks, however, is that, there is a large overhead involved in handling the frequent peer join and leave. Unlike structured P2P networks, the links between peers in an unstructured network are arbitrary and there is no relationship between the data and where a peer resides. The unstructured nature of the network makes searching a challenging problem. Many popular P2P networks such as Gnutella [1] and KaZaA [2] are unstructured P2P systems.

In this part of our dissertation, we present work related to each of our contributions: Search algorithms, knowledge representation schemes, topology optimization mechanisms and content replication methods for unstructured P2P networks.

2.1 Searching Method

Many techniques have been proposed by the research community for the purpose of efficient data locating. Searching in P2P networks can be broadly classified to *blind search*

and *informed search* methods.

2.1.1 Blind search methods

Blind search methods do not exploit any knowledge a peer has of its neighborhood to route queries through the P2P overlay. The most traditional and naive form of blind search techniques is flooding used by Gnutella [1]. This is a breadth-first-traversal of the P2P overlay graph. While this mechanism is simple to implement and requires no state maintained per node, it produces unacceptably large message overhead. In Random Walk [14], the query generator peer sends k query messages over randomly selected neighbors. Each message (also called a *walker*) terminates either when TTL (Time-To-Live) expires or when the required object is found. One of the most important advantages of Random Walk is the reduction in message cost ($k \times TTL$ in worst case) compared to that of flooding. However, this comes with the added disadvantage of variable performance based on the network topology and the random choice of peers. Modified BFS [15], is another variation of flooding, where peers randomly choose a ratio of its neighbors to forward a query. Even though this reduces message production compared to flooding, it still produces large number of messages. Iterative Depending [16] is yet another blind search mechanism that uses multiple breadth-first searches at successively larger depth limits until the query is satisfied or the maximum depth limit has been reached. This has lower search cost compared to flooding, but may result in massive amount of query messages being transmitted for some queries. Therefore, in certain situations, the search can produce even more messages than flooding.

2.1.2 Informed search methods

Informed search methods require peers to maintain some routing information that allows the queries to be forwarded intelligently to relevant peers. Therefore, compared to blind search approaches, informed search methods offer low message overhead and low latency. However, this is at the cost of maintaining various indices as local knowledge.

Works such as Routing Indices [17] , Adaptive Probabilistic Search (APS) [18] and Local Indices [16] use indexing mechanisms to efficiently route queries to qualified peers. In Routing Indices [17], each peer maintains indices which contain aggregate count of documents for predefined set of topics for its shared documents as well as for its neighbors. A goodness score is computed for each neighbor upon arrival of a topic based query, to find the best peers to forward it. Due to the intelligent peer selection mechanism, the bandwidth consumption of this search method is much better than flooding. Routing Indices introduce three indexing schemes: Compound Routing Index (CRI), Hop-count Routing Index (HRI) and Exponential Routing Index (ERI). The main disadvantage of CRI is that it does not take into account the hop count to destination documents into consideration. This problem is rectified by HRI, but at the cost of higher storage and transmission cost. ERI address these issues at the cost of potential loss in accuracy. Routing indices are much coarser than local indices which results in errors such as over-counting and under-counting. Also, the process of updating CRIs is very costly due to the need of propagating them along overlay paths. In Local Indices [19], each peer in the network maintains an index over the data of all nodes within r hops to itself. Therefore, a peer is capable of processing a query on behalf of peers up to r hops away from it. A system wide policy specifies depths at which a query should be processed. Only peers listed at given depths in policy process the query while others simply forward it. In this way, by processing data at fewer peers while not loosing information the search cost of the system is reduced. APS [18] is yet another intelligent search mechanism proposed for unstructured P2P networks. APS peers probabilistically forward future queries based on the responses they received from past queries. Each APS peer maintains an index which shows the relative probabilities of success of each of its neighbor for objects which it forwarded queries to. A query is deployed by sending out k walkers to those neighbors with highest relative probabilities to answer the query for that object. The relative probabilities of peers along a search path are increased (decreased) if the search walker succeeds (fails). The success rates and hits achieved in APS are considerably good compared to Random Walk and this performance improvement is achieved at same message complexity as Random Walk.

However, one of the drawbacks of APS is that peers discovered to be successful for certain objects are overloaded with search requests. Due to initial random selection of peers when no knowledge has been acquired of neighborhoods and future tendency to use the discovered successful peers, some peers with target objects do not get a chance to be discovered in a search process. Also, this method does not work well in finding rare objects.

Many of above mentioned search methods suffer from several drawbacks such as supporting only keyword-based searches and using large or static indices.

Semantic Searching To address the important issue of semantic based search in P2P systems, many solutions have been proposed over the past recent years. Semantic Overlay Networks (SON) [20] is one of the earliest works in the area. Here, authors propose a peer clustering approach based on semantic content a peer shares. Peers with similar semantic content are clustered together using virtual links with the aim of providing efficient query processing while preserving high degree of peer autonomy. Multiple SONs exist and queries are relayed to the most relevant SON and then flooded within that SON. Due to semantic clustering of peers, the search traffic is greatly reduced. This clustering methodology is completely distributed and also supports self-organization. However, this uses only simple predefined classification hierarchies for generation of SONs. A careful selection of classifications is required to obtain good system performance. Also, there is no guarantee that relevant peers are acquainted in finite time, especially in very large and dynamic P2P networks. In SON, peer joining process requires flooding the network to request classification hierarchy and also for finding semantically related peers, once its semantic interests are determined. The maintenance cost of SONs also grows with increase of peers in the network. Interest based shortcut [21] model takes a similar approach where peers with similar interests create shortcuts with one another with the aim of efficient content location. These shortcuts are generated and updated after each successful query and are used for guiding future requests. However, these interest based shortcuts are built on top of Gnutella overlay imposing the burden of maintaining virtual links on top of the original network overlay. Also, when these routing shortcuts fail or when no shortcuts exist, the peers need to resort to

flooding as the search mechanism. SETS [22] is yet another approach where authors propose a topic-segmentation based overlay for efficient query processing. The main drawback of SETS however, is that it relies on a single dedicated peer to cluster peers according to topic segments, requiring all other peers to contact it for cluster information. Also, the dedicated node can be ruined in presence of highly dynamic networks and frequent content updates. GES [23] addresses this single-point-of-failure problem in SETS by introducing a fully distributed cluster management mechanism. Here, peers are organized into semantic clusters based on the node clusters developed from document term vectors of peers. The queries are forwarded using biased walk through random long range links and thereafter flooded within a cluster using short range links once a relevant peer is found. GES introduces Local Data Clustering where each peer locally clusters its documents using data clustering techniques. Each cluster corresponds to a virtual node and a peer may host multiple virtual nodes who participate in topology adaptation and search. There are several drawbacks in GES: (1) the node vector representation may be inaccurate in presence of documents falling under multiple semantic categories, (2) the choice of long range links makes query routing less efficient due to not maintaining local state of peer clusters in the system. CSS [24] extends GES by presenting a class-based search system where all documents in a peer are clustered to different classes using a data clustering algorithm. Virtual short range and long range links are established based on class correlations between peers. Therefore CSS does not classify physical links like GES. Unlike in GES, a peer in CSS is required to maintain class vectors of its long range neighbors to intelligently route a query to relevant semantic clusters. During a search process, a query is routed through a intelligently selected long range link and flooded through short range links. This class based search is efficient because it uses a smaller search unit, a class of documents in a peer, instead of all documents in a peer. The major drawback of CSS however, is that it ignores the high cost of topology maintenance due to division of virtual nodes. pSearch [25] aims at constructing a semantic overlay on top of a m-dimensional CAN, a structured P2P network. The documents and queries are represented as semantic vectors using VSM and Latent Semantic Indexing (LSI) and are mapped

to semantic space using semantic vectors as keys. This is efficient in content location in terms of both latency and accuracy. However, LSI used for semantic vector calculation is very resource intensive, and its system performance under dynamic conditions is not known. pSearch stores the physical location of data objects in p places to address the issue of high dimensionality in semantic space, which makes space requirement and index publishing cost nontrivial. Authors of Semantic Small World (SSW) [26] introduce a novel overlay network and an index structure for semantic based P2P search to address problems in pSearch. The peers are dynamically clustered based on their document semantics and are organized into a small world network. Small worlds are characterized by high clustering coefficient and small search path length. SSW performs dimension reduction to construct a one dimensional SSW to address dimensionality issue in pSearch. SSW however, assumes homogeneous data in a peer which is unrealistic and also selects the largest cluster centroid as a peers semantic representation.

Several semantic based indexing methods have also been proposed by the research community. Ontology based Local Index (OLI) [27] is one such scheme where an ontology based index method for P2P networks is proposed. Implementation is provided for the structured network HyperCup [28] to reduce network traffic. The peers assume a globally shared ontology and the concepts in the ontology are used as reference to build indexes. Limitations of OLI include, not considering number of documents or information content of each concept, hop distance to documents etc. Also, since it assumes a previously available routing algorithm, such as one available in HyperCup, OLI cannot be applied for unstructured networks. Semantic search methods applied in structured networks 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. The authors of [29] introduce an ontology based routing index for unstructured P2P networks to resolve these issues in OLI. A matching function which takes into account number of documents accessible via a link is used to rank and select neighbors for query forwarding. Both OLI and [29] assume a global ontology which is an unrealistic assumption and also exploits

only inheritance relation in ontology for query routing. There are also P2P database related methods such as RDFPeers [30], XP2P [31] and PeerDB [32] proposed in the recent past. The drawback associated with these approaches however is that they require the end user to specify query in a database query language such as SQL.

Table 2.1 summarizes some popular search techniques in P2P domain.

2.2 Knowledge Representation

A good knowledge representation in peers is crucial for a successful search technique to perform query routing and local query evaluation. The quality of the query routing directly depends on the efficiency of the knowledge representation at a peer of its neighborhood as this information is exploited in selecting neighbors for query routing. Many people have used Indexing [16], Vector Space Model (VSM) [33] and Bloom filters [34] to represent and summarize entities of interest in P2P networks. Below we describe popular knowledge presentation techniques for each category.

2.2.1 Indexing

Index engineering is at the heart of P2P search methods. P2P index can be either a local, a centralized or a distributed.

Centralized indexing: Centralized indexing typically rely on a unique entity in the network that stores the index. Peers have to query the central index in order to know where the content of interest is stored. Classic examples of systems adopting centralized indexing include Napster [35] and iMesh [36]. Several Recent approaches combine global knowledge with local indexes have also been proposed. For example, PlanetP [37] collects information regarding other network peers' shared documents via gossip in an unstructured P2P network thus maintaining a local copy of the global directory. The system appears to be limited to a few thousand peers. MINERVA [38] is yet another P2P system where maintains a global index with peer collection statistics in a structured P2P overlay to facilitate the peer selection process. MINERVA builds a process where it penalizes peers holding overlapping document

collections.

Local indexing: A Local indexing allows peers to index of its own content. Therefore during query routing process each query receiver peer checks its local index for existence of requested content. This has been used by unstructured P2P networks such as Gnutella [1].In unstructured networks with super-peers, typically super-peers keep an index for the ordinary peers attached to them.

Distributed Indexing: With distributed indexes, peers maintain pointers towards the target objects that reside in some other peers. Distributed indexes are used in most P2P designs nowadays . Popular works such as Routing Indices [17],APS [18], OLI) [27] etc. Routing Indeices [17] maintain goodness of neighbors for each content topic using the number of documents while APS [18] maintains a relative probability of success of each neighbor per object it has been queried before. Odissea [39] is yet another distributed indexing mechanism which assumes a two-layered search engine architecture with a global distributed index structure distributed over the nodes in the system. A single node holds the complete, Web-scale, index for a given text term .

2.2.2 Vector Space Model (VSM)

VSMs) [33] have several attractive properties. VSMs extract knowledge automatically from a given corpus, thus they require much less labor than other approaches to semantics, such as hand-coded knowledge bases and ontologies. They are simple to maintain and update, an attractive property of any knowledge representation structure suitable for dynamic distributed environments. VSM have been used by the P2P research community for representing various abstract entities such as documents, peers and even peer clusters due to their simplicity and effectiveness in information quantification as well as similarity measurement. Popular works towards this direction include [22], [23] , [24], [40] and [41].

2.2.3 Bloom Filters

Bloom Filters [34] are bit arrays for representing set of elements. They are well known for their efficiency in representing large volume data sets in a compact and space efficient manner and supporting membership queries over the represented data. This dramatic reduction of space consumption in Bloom filters come at the cost of introducing false positives. Therefore in P2P networks where space and communication bandwidth is at premium Bloom Filters offer attractive low cost knowledge representation and dissemination mechanism at a controllable false positive rate. Bloom filters have been used in early Web Cache Sharing systems [42] where proxies represent their known urls using Bloom filters and broadcasted to other proxies in a space efficient manner for knowledge sharing. Bloom filters also found very early uses in databases [43], [44]. In the contenxt of P2P, Bloom filters have been used extensively for both compact information representation and transmission [45], [37], [46], [47], [48], [49], [50]. PlanetP [37] use Bloom Filters to store keywords associated with documents.Reference [46] introduces a new data structure called *Approximate reconciliation trees* that uses Bloom filters. The approximate reconciliation trees use Bloom filters on top of a tree structure, which are used in cryptographic settings to minimize the amount of data transmitted for verification. Authors of [47] show how an extension of traditional Bloom filters, called multi-level Bloom filters, can be used to route path queries in a P2P system where peers store and query for XML documents. The hierarchical XML documents are encoded in either a *Breadth Bloom Filter* where every element in a single level of a XML hierarchy is represented by a separate Bloom filter, or in a *Depth Bloom Filter* where paths of the same length are represented as a separate Bloom filter. Authors further introduce a mechanism where Bloom filters are be used to build content-based overlay networks where peers with similar content are link together in the overlay.The similarity of the content of two peers is defined based on the similarity of their filters. A new extension to traditional Bloom filter called *Exponential Decay Bloom Filter (EDBF)* is introduced in [48]. EDBF is capable of encoding probabilistic information useful for query routing.

2.2.4 Semantic Models

While many knowledge representation models have been proposed in the past, majority of these methods however often overlook the need for providing semantically relevant information. One of the most primitive forms of semantic meta-data representation is taxonomies. Taxonomy is a classification of concepts based on inheritance. However taxonomies are too simple to represent complexity of relationships in real life. Using a thesaurus is another popular method used in semantic data models. Compared to taxonomy, thesaurus extends taxonomy by introducing new relationships for similarity and synonymy. Both taxonomies and thesaurus however suffer from the drawback of not being able to uniquely identify terms or instances (redundancy) as they allow terms to appear more than once in a taxonomy based on inheritance relation. Also, a thesaurus is incapable of distinguishing between homonyms. Another semantic model proposed is topic map, which is a classification of concepts. In addition to inheritance, similarity and synonymy relationships, it also allows user-defined relationships to be specified. However, the problem of redundancy is not addressed in topic maps either. Ontology is defined to be most complete and powerful model for information representation. Ontology is richer than any other semantic model in that it allows classification of not only concepts but also their instances. One can also define complex relationships and restrictions using an ontology.Popular semantic technologies used by the research community today include XML, RDF and OWL.

While semantic meta-data models have been proposed extensively in the research community, most works to date lack many desired characteristics expected from semantic data representation models suitable for completely decentralized and dynamic P2P environments. A semantic metadata model optimized for P2P environments would exhibit characteristics such as (i) ability to encode past domain experience as knowledge in a quantitative manner (ii) simplicity of model (iii) low dimensionality (iv) low-overhead maintenance and (v) ability to make probabilistic inferences on semantic data. While classical TF-IDF scheme based information quantification methods used in Vector Space Models (VSMs) are simple and straightforward, they lack in semantic expressiveness and suffer from large dimensionality of

the resulting term vectors which is computationally expensive. More sophisticated semantic representation models such as latent semantic indexing [51] are too resource consuming to construct and maintain in dynamic P2P networks and there are no proper guidelines to determine the number of dimensions for Singular Value Decomposition (SVD).

Use of ontologies to overcome the limitations of existing knowledge representation methods has been popular since the emergence of semantic web. While there have been many contributions in this direction over the last few years [52], [53], [54], many do not use the full potential of expressive capability of ontologies or are based on Boolean retrieval models. As surprising as it may sound, ontologies are a shallow form of semantic representation as long as they are not integrated with proper knowledge quantification methodologies and appropriate quantitative reasoning techniques. Massive amounts of information available in web today are largely unstructured. Converting such a huge amount of heterogeneous unstructured data distributed over a network at an affordable cost is a problem yet to be solved. The knowledge representation data structure devised for P2P environments should also provide probabilistic reasoning to be acted upon them for query routing purposes. Probabilistic reasoning is a well-understood and theoretically-sound method for combining information from varying data sources with varying reliability. It has shown its applicability across many applications including query routing in P2P networks. However, only limited amount of work has been done to combine probabilistic reasoning capabilities with ontologies [55]. Heterogeneity of data and ontologies in distributed networks poses further challenges to the problem of probabilistic reasoning in P2P networks. While probability theory provides the pathway to produce solutions that indicate the degree of plausibility, ontologies use logical reasoning which provides only definite answers.

Following table summarizes some of the popular knowledge representation techniques, their advantages and disadvantages.

2.3 Topology Optimization

Topology adaptation is a widely used approach to enhance search performance in P2P networks. There have been multiple studies on how to enhance search performance by changing P2P topology. Authors of [59] introduce a mechanism for building and maintaining the square-root topology. However, authors make the unrealistic assumptions that each data type has only one replica and there is no limit on hop count. Reference [60] . Most existing works on topology adaptation are based on the heterogeneous features on P2P networks. Physical heterogeneity in P2P networks are measured using parameters such as network bandwidth, peers' processing capacity and distance between peers. Works on this aspect include [61], [62], [63], [64] and [65]. Topology optimization based on peers similarity of interests has also been a popular mechanism of reorganizing the network structure. To this end, works such as [66], have been proposed to organize the P2P overlay so that peers with similar content are physically grouped together. In PROSA [66] , peer may establish three types of links: Fully Semantic Link, Temporary Semantic Link or Acquaintance Link according to the degree of knowing each other. The works [20], [67], [22] and [23] build a semantic overlay by establishing links among most semantic similar nodes.Query routing is first done by global routing through long links which are established between semantic dissimilar nodes, and then does flooding by local routing through short links which are established between semantic similar peers.Works such as [68], [69] and [70] optimize the original P2P topology based on small world phenomenon [71]. The notion of small-world phenomenon originated from social science research. Recent studies show that peer-to-peer networks like Freenet may exhibit the small-world properties [72].

2.4 Content Replication

Replication is a popular mechanism used to achieve high availability and improved search performance in P2P networks. A good replication mechanism should intelligently decide on what items need to be replicated, where and when so that the overhead introduced

on network by replication is minimal. Mechanisms such as owner replication, path replication and random replication [14], [73], [74], are based on site selection policy for replicating an object found by a query. The owner replication replicates an object only at the requesting peer. In path based replication mechanism [14] the replicas are placed in all peers in along the path a query traveled. A variation of this called *path random replication* is introduced in [75]. Each intermediate peer randomly determines whether or not the replica is created and placed there, based on the probability of the pre-determined replication ratio. With Push-the-pull replication [76], after a successful search, the requesting node enters a replicate-push phase where it transmits copies of the item to its neighbors in order to obtain square root replication. Adaptive Probabilistic Replication (APRE) [77] is yet another distributed protocol that automatically fine-tunes the replication ratio of each shared object according to the current demand for it. APRE offers a direct response to workload changes. Reference [78] introduce an adaptive, fully distributed technique that dynamically replicates content in a near-optimal manner. The optimal object replication includes a logarithmic assignment rule, which provides a closed-form optimal solution to the continuous approximation of the problem.

Table 2.3 summarizes some popular replication techniques.

Table (2.1) Comparison of P2P Search Techniques

Search algorithm	**Network Type**	**Search Type**	**Query Processing Strategy**	**Commments**
Gnutella [1]	Unstructured	Blind	Flooding queries to whole neighborhood with fixed TTL	High Recall at small TTLs, Large message overhead, Low scalability
Random Walk [14]	Unstructured	Blind	K nods selected randomly to forward query with fixed TTL	Low message overhead than flooding, Variable performance with network topology and random choice
Routing Indices [17]	Unstructured	Informed (Indexing)	The best set of neighbors are selected based on a goodness score computed based on routing indices	Low bandwidth consumption, High updating cost of indices, Over-counting and under-counting errors due to summarization
SON [20]	Unstructured	Informed (Clustering)	The query is first forwarded to the most suited SON and then flooded within the SON	High recall at low message cost, No guarantee provided that relevant peers acquired in finite time
pSearch [25]	Structured	Informed (Clustering)	A query is routed through CAN based on its semantics to relevant peer. Upon reaching destination, query is flooded within radius r	High recall, LSI is computationally expensive
SSW [26]	Unstructured	Informed (Semantic clustering and indexing)	A search query is evaluated against current cluster by flooding query within neighborhood of query receiver peer if its cluster qualifies to answer query	Scalable to very large network sizes and very large number of data objects, Introduce attractive tradeoff between search path length and maintenance cost due to Small world, Assumes homogeneous data within a peer

Table (2.2) Comparison of Knowledge Representation Techniques

Technique	**P2P Releted Work**	**Advantages**	**Disadvantages**
Document Identifier	[18], [21]	Simple to implement	Difficult to encode document semantics
VSM	[19], [22]	Simple to implement, Allows easy evaluation of queries against documents using cosine similarity	semantic relations between terms cannot be represented, Term co-occurrences in document and order of terms is lost
Latent Semantic Indexing (LSI)	[25], [26]	Resolves synonymy and polysemy issues, Strictly mathematical structure requires no thesauri or dictionary	High computational cost and high memory consumption, Difficult to determine number of dimensions for singular value decomposition
Bloom Filters	[48], [37]	Compact data representation, Constant time required to add or check an item in a set, regardless of number of elements in the set	False positives are possible, in presence of large number of elements in a set
XML	[56], [57]	Easily encode objects, their properties and relations, Query languages are readily available for XML data, Custom tags can be defined by user	Unstructured documents cannot be automatically converted to XML format. Manual intervention required, query languages for XML such as XQuery which are complex and not end-user friendly
RDF	[30], [58]	Allows inference to contextually broaden search, retrieval and analysis, Allows semantic based representation and queries	Unstructured documents cannot be automatically converted to RDF format. Manual intervention required
Ontology(E.g. OWL)	[27], [29]	Allows inference to contextually broaden search, retrieval and analysis, Provides richer properties, additional class restrictions, relationships than RDF	User needs to learn query languages such as SPARQL, OWLQL which are complex and not end-user friendly

Table (2.3) Comparison of Replication Techniques

Replication Technique	**Strategy of replication**	**Advantages**	**Disadvantages**
Uniform Replication	Replicates all objects uniformly in the network	All resources are equally replicated regardless of their popularity	replicas may be unnecessarily replicated
Square Root Replication	The number of replicas of a file is proportional to the square-root of query distribution	Optimal replication	The search performance relies on the selection of suitable sites for hosting new replicas
Owner Replication	The object is replicated only at the requester node once the file is found	The number of replicas is proportion to the number of requests	Takes a large amount of time to propagate replicas over the P2P network
Random Replication	Replicate object in randomly selected peers	Creates the same number of replicas as of path replication	The peer require global knowledge of the existence of all the peers in overlay
Path Replication	Stores the object along the path of a successful query walker	simple to implement	creates large number of replicas

PART 3

REPLICATION BASED RARE OBJECT LOCATION

Resource management and search is very important yet challenging in large-scale distributed systems like P2P networks. While many work have been proposed for search in P2P networks, the problem of efficient object discovery especially for rare objects has not been addressed adequately.

Unstructured P2P networks [1, 2] do not impose any such constraints resulting in peers exercising full autonomy over their data and overlay links they establish. These networks are arguably more resilient to peer population transiency. Being large-scale unsupervised environments, however, unstructured P2P networks often suffer from low search efficiency and high search traffic. The poor search efficiency of unstructured P2P networks is especially observed in search for rare objects. The problem of search for unstructured P2P networks has been addressed by many previous work such as flooding, expanding ring [14], random walk [79] and Adaptive Probabilistic Search [80]. While most of these algorithms are very effective in locating popular content, they fall short in locating rare content. As shown in [10], 18% of queries return no responses even when the target objects exist in the overlay network.

Often in real world P2P networks the popularity of content (measured with respect to the number of object replicas distributed in the network) are not uniform but skewed. Studies have shown that web requests on the internet space follows a Zip-like distribution [81]. Therefore, in the reality, this skewed popularity distribution will result in unbalanced load among peers heavily loading those peers highly rich in searched content. Therefor a popularity estimation techniques for a unpredictable environments with transient populations such as P2P overlays should be capable of dynamically estimating popularity of documents. While due to high number of replicas distributed, finding a popular object in unstructured

P2P networks can be performed efficiently using simple indexing strategies, in finding a rare object, the search scope should be extended to increase the probability of locating the object. Such a solution, however, results is unacceptably high search traffic.

In this thesis, we propose a simple, yet powerful probabilistic indexing mechanism called R-SPUN to increase efficiency of search in unstructured P2P networks. The indexing mechanism utilizes the phenomena of emergence of paths that successful queries travel and peers' degree of connectivity to those paths to derive probabilistic conclusions regarding success strength of each peer for requested objects. In combination with the indexing strategy, we present a novel rare object replication mechanism that utilize the graph property of Connected Dominating Sets(CDS) for rare object replication. We also present a resource discovery algorithm where the peer selection decision in query routing is based on the popularity of the requested object as well as connectivity of the peer to the successful query paths developed over time in the network. To achieve small index sizes, we limit the index size of a peer based on its storage capacity. We propose a Bloom Filter based strategy for a peer to keep track of the set of documents reachable in a given hop radius that compensates for the knowledge loss due to bounding index sizes. Performance evaluation demonstrates that our proposed approach can dramatically improve the search efficiency of unstructured P2P systems while keeping the communication cost at a level comparable with the state-of-art unstructured P2P systems.

Efficient object popularity estimation is a crucial component in object replication algorithms. We utilize Bloom filters as well as the *Connected Dominating Set*(CDS) of the network to estimate the global popularity of distributed objects in the network. Bloom filters are space efficient probabilistic data structures that can encode large volumes of data. CDS of a network is a connected subset of peers that are ever other peer not in CDS is connected to. Thus utilizing CDS as a communication backbone and Bloom filters as data transfer medium results in low communication cost of popularity estimation and replication. The rare objects are replicated along the CDS of the overlay so that any peer can reach rare objects in a few hops. The search requests are forwarded to the peers more likely to contain

target objects by exploiting peers' local knowledge of the popularity of the requested object as well as the relevance strength of their neighbors in terms of the degree of connectivity to the previously discovered query paths neighbors lead to. Simulations show that our indexing and replication strategy greatly improves the search efficiency at moderate search cost regardless of the popularity of the search object.

The rest of the chapter is organized as follows. In Section 4.1 we present the preliminaries of our work. In Section 3.2 we present our novel idea of evolution of successful query paths and associated data structures. We present our indexing scheme in Section 3.3 followed by our replication strategy in Section 3.4. Section 6.3.3 presents our query routing algorithm. In Section 4.6 we summarize our contributions and conclude the chapter.

3.1 Preliminaries

3.1.1 System Model

In R-SPUN we put forward a controlled flooding algorithm, where each peer probabilistically forwards queries to its neighbors based on the popularity of objects requested. Our model framework is defined for an unstructured P2P network where peers launch queries for various objects distributed across the network. A search process is initiated when a peer launches a query to find a specific object by deploying k walkers. A walker is a duplicate of the original query. Once deployed, a walker continues traveling the network along its own path until the requested object is found or until Time-To-Live (TTL), an upper bound on number of hops, expires, in which case the walker terminates. If the requested object is not found until TTL has expired, a miss is returned along the return path of the query. Otherwise, a hit is returned in the reverse query path. This returned hit or miss information is used by each intermediate peer along the query path (including the querying peer) to update its routing index's relative probability of success assigned to its neighbor who sent hit or miss as well as to determine object popularity in its local vicinity. For example, if a search walker is generated at peer A and sent along the path $A \rightarrow B \rightarrow C \rightarrow D$, the hit or

miss will be returned in the path $D \rightarrow C \rightarrow B \rightarrow A$. The information carried by hit or miss message is used by C to update its local knowledge about D, B to update its local knowledge about C, and finally A to update its local knowledge about B. A search query terminates when all the deployed walkers have terminated either with a success or a failure.

3.1.2 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. There are many variants of standard BF such as Spectral BF, counting BF, compressed BF, etc. In our work we utilize 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. They allow querying on item multiplicities as well as deletion.

3.1.3 Connected Dominating Sets

A *Dominating Set* of a graph is a subset of nodes such that every node not in the DS is adjacent to at least one member of DS. A *Connected Dominating Set* (CDS) is a DS where sub-graph induced by DS is connected. Therefore a CDS of a P2P network is a connected subset of nodes of the network from which all nodes in the network can be reached in one-hop. Finding a minimum CDS is NP-complete for most graphs. Authors of [82] put forward a marking process based distributed algorithm for calculating CDS.

3.1.4 Reinforcement Learning

Reinforcement Learning(RL) addresses the general issue of how a learner that interacts with its environment can learn the optimal actions to achieve its goal. RL is characterized

by the trial-and-error learning and the delayed reward mechanism. In each step the agent chooses an action for the state of the environments. Whenever a learner takes an action, it receives an immediate reward and the environment changes its state. Depending on the reward and the latest states, agent chooses the next action to increase the probability of the plus rewards. RL has been proven in artificial intelligence to be able to learn the best sequence of actions in order to achieve a certain goal.

3.2 System Architecture

Existing unstructured networks have one main problem: they are highly limited in their ability to locate rare items. The goal of our work is to develop techniques that that perform efficient search regardless of the popularity of the objects in unstructured networks.

Four main techniques lie at the core of our design:

1. Successful query path evolution based indexing
2. Bloom Filter based knowledge representation and dissemination
3. Rare object replication and
4. Connected Dominating Set assisted query routing

The main intuition behind them is to guide future queries along successful query paths previously discovered. To ensure that queries terminate with high success rate, we need to efficiently replicate rare objects at sufficiently close distance from peers . Our routing index, Bloom Filter based knowledge representation and search algorithms address the former while our local object popularity estimation and CDS based replication achieves the latter.

Next we describe the core concept of evolution of successful query paths that our searching framework is built around.

3.2.1 Evolution of Successful Query Paths

Peers in the P2P network launch queries for various objects during their lifetime. Understandably, peers at various physical locations in the network may be interested in finding

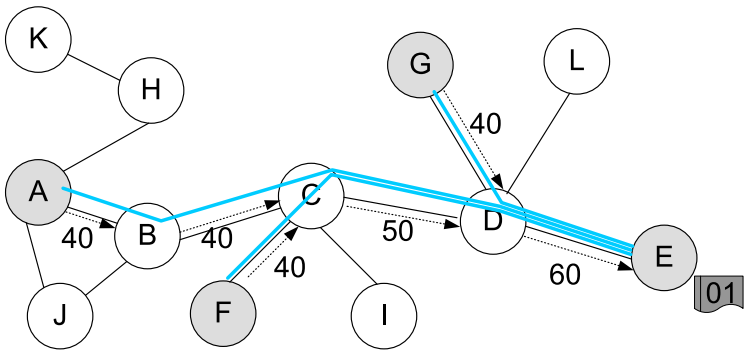


Figure (3.1) An example of successful query path development. The values on edges show the relative probability of success value assigned to a peer by its neighbor. The arrow shows the direction of assignment.

some popular objects. Multiple replicas of such objects may be distributed among different peers in the network. When various peers launch search walkers for the same object, some walkers succeed at finding the requested object. In this way, many successful query paths develop in the P2P network over time. We call a query path that a successful walker travels a *successful query path*. R-SPUN algorithm track successful query paths as these develop. Fig.3.1 shows an example of three success query paths for a given object 01: A→B→C→D→E, F→C→D→E and G→D→E in a twelve node P2P network developed as a result of successful queries generated by A, F and G for object 01.

We propose to use reinforcement learning (RL)to catalyze emergence and reuse of discovered successful query paths. In R-SPUN, we employ RL by allowing each intermediate peer P in a query path to assign a relative probability value to its neighbor in a query path, farther away from query originator peer. This relative probability value represents how successful the neighbor is at finding the queried object compared to P's other neighbors and the query is routed through the neighbor(s) with highest relative probability values(s). The values on edges in Fig. 3.1 are these relative probabilities of success assigned by one peer to another. For example, the value 40 on edge from A to B represents the relative probability of success (i.e., *success ratio*) assigned by A to B for object 01.

Intuitively, multiple successful query paths for a given object may intersect at some peer and join to form a single path from that peer to the destination peer containing the target

object. For example, in Fig.3.1, the two query paths starting at A and F intersect at C to form a single successful query path C→D→E. This phenomenon occurs as a result of the associated positive feedback loop with the (re)use of successful query path so that peers who forward a query along a successful query path receive an immediately reward thus reinforcing the successful query path. Employment of a successful query path to deploy a search walker results in higher probability of hits. This in turn contributes to peers increasing the relative probabilities of success assigned to their neighbors along such query paths. Therefore, due to reuse, the success ratios in a successful query path monotonically increase from querying peer towards the destination peer. In R-SPUN, peers identify these successful query paths using *Best Path Gradient* (BPG) criterion which will be introduced in section 6.3.3.

The intelligent neighbor selection process in R-SPUN is assisted by four different data structures as described in the section 3.2.2.

3.2.2 Local Knowledge of a Peer

Here we describe the local data structures a peer maintains for intelligent query routing.

Routing Index (R-SPUN Index)

R-SPUN requires each peer to keep a local index of relative probabilities of success per neighbor, for each object it requested through that neighbor. We call these relative probabilities of success values, the *Relative Success Ratios (RSRs)*, and the local index, the *R-SPUN index*. Each peer in R-SPUN keeps a vector of relative success ratios per neighbor for each object it has requested for. Each element of a given vector represents the relative probability of success at certain hops away from the neighbor. The formal definition of a *Relative Success Ratio Vector (RSRV)* is given below:

Definition 3.1. *Relative Success Ratio Vector*:

The Relative Success Ratio Vector (RSRV) of a given neighbor P in SPUN index of a peer

for object o is denoted as ($RSRV_P^o$) and a RSR value in vector is defined as follows:

$$RSRV_P^o[h] = \frac{Relative\ success\ ratio\ at\ h\ hops}{from\ neighbor\ P\ for\ object\ o} \tag{3.1}$$

SPUN Index (A):

Object ID	Neighbor	RSR_0	RSR_1	RSR_2	RSR_3
01	J	30			
01	B	40	50	60	60
01	H	40	60		
05	J	20	20	20	10
05	B	40	40	40	
05	H	30			

$RSRV_B^{01}$

Figure (3.2) Example SPUN Index of peer A in Figure 3.1. The RSR_i in A's SPUN index stands for the relative success ratio i hops away from considered neighbor for a given object.

The objective of our search process is to locate at least one document containing the queried object. A search walker therefore terminates either once the requested object is found or its TTL expires. Therefore the distance a search walker travels might vary from one hop up to TTL hops. Therefore, the vectors maintained per peer for an object may be of different lengths Fig.3.2 shows an example R-SPUN index for peer A in Fig.3.1. The RSR_i denotes the relative success ratio i hops away from A through a given neighbor for a given object. For instance, the value 40 in the second entry under column RSR_0 represents the RSR of neighbor B at zero hop distance from A , value 50 in under column RSR_1 represents the RSR of B at one hop distance from A and so on. The figure also highlights the $RSRV$ of neighbor B for object 01. How these entries in R-SPUN indices are created and updated will be discussed in detail under section 3.3.1.

Reachable Documents Bloom Filter

One of the major restrictions to observe in devising indexing structures is storage capacity. Given a limited storage space we need to limit index size to maintain smaller indices. Therefore completely relying in the routing index results in the unavoidable possibility of misses even when the desired object is at a reachable distance to a querying peer. This will degrade the performance of our search process. Any other mechanism used for compensating for this degrade in performance should not consume unacceptably large storage and communication cost. To overcome this problem, we use Bloom Filters to represent the set of documents reachable by a peer within a given hop radius. Bloom filters are well know for its space efficient representation. The Bloom filter maintenance mechanism does not incur any additional overhead of communication cost as it relies on the same information that is used for updating the routing index. Let B_P be the Bloom filter of peer P. Each peer P in R-SPUN disseminate B_P to all peers within a TTL hop radius, so that every peer within the TTL radius limit can answer queries about whether P might have a particular document one is looking for even when the queried document is not recorded in its routing index.

Peer Profile

A peers profile consists of its latest history of the queries either generated by or forwarded through it and is formally defined below.

Definition 2. *Peer Profile*

The profile of a peer P (PP_P) can be represented by a set whose elements are a pair: queried object ID and associated RSR:

$$PP_P = \{(o, r) | o : objectID,\ r : relative\, success\, ratio\, of\, o\} \tag{3.2}$$

If a hit or miss is generated by the peer itself in response to a query for an object o, the filed RSR contains a Boolean value stating whether it is a *hit* or a *miss*. Otherwise, RSR contains the RSR_0 for the neighbor associated with the latest query for o that passed through P. For example, consider a profile of a peer B (PP_B) in Fig.3.3(b) (05,60), (08,MISS). The

RSR_0 Indices	Object ID	RSR_0 before update	RSR_0 after update
RSR_0(A→ B)	05	50	60
RSR_0(A→ B)	08	-	10
RSR_0(B→ C)	05	40	70
RSR_0(C→ D)	03	40	50

(a)

A→B→C→D:

PP_D = {(03,HIT)}

PP_C = {(05,70)}

PP_B = {(05,60),(08,MISS)}

(b)

Figure (3.3) (a) SPUN Index before and after updated by received neighbor profiles. (b) Peer Profiles received by peers in query path $A \rightarrow B \rightarrow C \rightarrow D$

RSR_i Indices	Object ID	RSR_i before update	RSR_i after update
RSR_1(A→ B)	20	-	40
RSR_2(A→ B)	20	40	50
RSR_2(B→ C)	20	30	50
RSR_1(C→ D)	20	30	20

(a)

A→B→C→D:

$RSRV_B^{20}$ = [40,50]

$RSRV_C^{20}$ = [50]

A→F→G:

$RSRV_F^{20}$ = [20]

(b)

Figure (3.4) (a) SPUN Index before and after update by RSRVs received in messages. (b) The RSRVs received by peers in two query paths.

first element means that B has seen a hit or miss for object 05 recently and the RSR_0 value for object 05 is 60 for B's neighbor associated with the query. The second element denotes B generated a miss for a recent query for object 08.

3.2.3 Message Format

Fig. 3.5 shows the message format ued in R-SPUN protocol. The sequence number allows the message to be uniquely identified. During query transition TTL decrease at every hop. While the size of the visited peer list size increases as the message travels more hops, it is bounded by the TTL value. Similarly, the RSRV vector size and number Of Bloom Filters in a message are also bounded by the same parameter TTL. A message transmits one peer profile at a time and its size is fixed by a system defined parameter. Therefore, peer profile does not have an effect on the message payload size.

Length (bytes)	Field
4	Sequence Number
1	Payload Descriptor (e.g. Query)
1	TTL
1	Message Sender Peer ID
1	Search Object ID
1	Search Mode (e.g. Regular, CDS)
Variable	Visited Peer List
20	Peer Profile of Message Sender Peer
Variable	RSRVs of Visited Peer List

Figure (3.5) Example SPUN Index of peer A in Figure 3.1. The RSR_i in A's SPUN index stands for the relative success ratio i hops away from considered neighbor for a given object.

3.3 Local knowledge Construction and Maintenance

In this section we discuss how the local knowledge of a peer is created and updated.

3.3.1 Routing Index Construction and Maintenance

R-SPUN routing index is created and updated on the fly when peers discover documents reachable through its neighbors. The R-SPUN index of a peer is updated in two ways: (i) Proactive update before sending a query, (ii) Reactive update based on return message

Proactive Update

Whatever is the peer selection criterion used, each intermediate peer in a query path including querying peer, updates the RSR_0 values of the selected neighbors for the queried object in its local R-SPUN index prior to sending or forwarding a query. In case there are no entries for the queried object in a query sending (or forwarding) peers R-SPUN index, the peer will create an entry for the queried object for each of its neighbors, by placing a default RSR value of 30 in RSR_0 field in each new entry. Placing the same RSR value indicates that all peers have the same probability of success for the query. If entries already existed for the queried object in peers local R-SPUN index, however, the peer will proactively increase the RSR_0 value of the entries for the querying object of selected neighbors by a default value of 10, before launching or forwarding the query assuming the query will succeed. This increased

index value denotes higher relative probability of success in discovering that object through the given neighbor compared to other neighbors.

Reactive Update A reactive update of index occurs when a peer recieves a hit or a miss message for a query it transmitted. The update is carried out based on the information encapsulated in the recived return message. Based on the types of information, a peer perform three updates:

- *Update based on return message type*

  The proactive increments discussed above are performed by peers in a query path assuming the walker will succeed. If the walker actually returns a hit, no further changes need to be made to their SPUN indices. If the walker returns a miss, however, each miss receiver peer will decrement the RSR_0 values of the neighbor who sent the miss for the queried object by a default value of 20 to reflect that the neighbor was unsuccessful in finding the object. We select the post decrement value (i.e., 20) to be greater than proactive increment (i.e., 10) to make the relative probability of success of the unsuccessful peers for the queried object to be less than (or equal to) other neighbors.

  Fig.3.4(b) shows the RSR_0 values of R-SPUN indices of peers involved in two walkers deployed by A shown in Fig.6.2 for the queried object. Both the RSR_0 values for before and after update with hit or miss information are also shown. The $RSR_0(X \rightarrow Y)$ in the figure denotes the RSR_0 value of neighbor Y in Xs R-SPUN index for the queried object. As Fig.3.4(b) shows, the proactive update of R-SPUN index values of $RSR_0(A \rightarrow F)$ and $RSR_0(F \rightarrow G)$ are decremented from 40 (i.e., proactive increment) to 20 as walker through F and G eventually failed while $RSR_0(A \rightarrow B)$, $RSR_0(B \rightarrow C)$ and $RSR_0(C \rightarrow D)$ remained 40 as walkers through path $A \rightarrow B \rightarrow C$ succeeded.

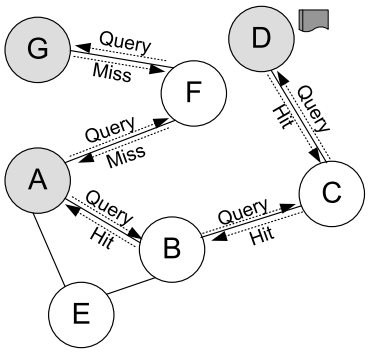
- *Update based on RSRVs in return message*

Each return message, whether a hit or a miss, contains a vector of relative success ratio values ($RSRV$). This $RSRV$ contains the relative success ratios that have been appended by the intermediate peers in the query path that the message has visited so far along the query path.

For example, if a query is sent along path $A \rightarrow B \rightarrow C \rightarrow D$) for object 01, the hit or miss message A receives from B contains a vector of relative success ratios consisting of two elements: RSR_0 value of D for object 01 in C's R-SPUN index and RSR_0 value of C for object 01 in B's R-SPUN index. Each intermediate peer in a query path (including querying peer) updates its R-SPUN index using this $RSRV$ of a hit or miss message query. Fig.3.4(b) shows set of $RSRV$s returned in two query paths $A \rightarrow F \rightarrow G$) and $A \rightarrow B \rightarrow C \rightarrow D$). In the figure, a $RSRV$ received by a peer X from a neighbor P for object o is denoted by $RSRV_P^o$. For example, the $RSRV$ peer A receives from peer B for an object o, o B $RSRV$ =[40, 50]. The information in o B $RSRV$ carried by the return message is the most recent information A has for B. Therefore these values are inserted at relevant RSR_i fields of B's entry in A's R-SPUN for the given object, replacing any existing stale values. For example, value 40 in o B $RSRV$ is assigned to RSR_1 in As SPUN index while value 50 replaces the old RSR_2 value 40 in neighbor Bs entry for object o.

- *Update based on Peer Profile of the sender of the message*

  Unlike for $RSRV$s which require a hit or miss message to be carried, peer profiles can be piggybacked in any type of message. Therefore this update technique occurs at receipt of message of any type at a peer. A peer profile travels only one hop distance. Each peer currently visited by a message makes use of the peer profile sent by its neighbor to update its R-SPUN index. A peer also appends its own profile to messages generated or forwarded through it. For each queried object listed in a neighbor's profile, receiver of the neighbor's profile updates the corresponding entry of its R-SPUN index. If an entry in the neighbor's profile has the binary information HIT or MISS associated with



(a)

RSR_0 Indices	Object ID	Initial RSR_0	RSR_0 before update	RSR_0 after update
RSR_0(A→ B)	20	30	40	40
RSR_0(B→ C)	20	30	40	40
RSR_0(C→ D)	20	30	40	40
RSR_0(A→ F)	20	30	40	20
RSR_0(F→ G)	20	30	40	20
RSR_0(A→ E)	20	30	30	30
RSR_0(B→ E)	20	30	30	30

(b)

Figure (3.6) (a) Query for an object stored at peer D. Two search walkers are deployed by A. (b) R-SPUN Index before and after update by Hit or Miss Information. $X \rightarrow Y$ denotes the RSR_0 value of peer Y stored at SPUN index of peer X for the queried object.

an object, the RSR_0 value of neighbor for queried object in receiver peer's R-SPUN index is incremented or decremented respectively, according to APS criterion. If the entry in neighbor's profile contains an actual relative success ratio, the current RSR_0 value of the relevant entry in receiver peer's R-SPUN index is replaced by this latest information. Fig.3.3(b) shows the profiles of peers that are sent to their neighbors along the reverse query path and Fig.3.6 show RSR_0 values before and after updating corresponding R-SPUN indices using these peer profiles.

These R-SPUN index updates take places at every intermediate peer in a query path a walker travels, including the querying peer. The algorithm for SPUN index update of a peer based on return message is given in Algorithm 3.1.

3.3.2 Bloom Filter based Knowledge Dissemination and Update

As mentioned in section 3.1.2, bloom filters can be efficiently used to represent large document collection in compact manner. Objective of a peers P's bloom filter B_P is to allow P to acquire knowledge of the set of all the documents reachable through each neighbor within a TTL radius. To acquire such information at lower communication cos we propose the following method:

Algorithm 3.1 *R-SPUN Index Update*

Input:

Mes = Message
P = Message receiver peer
N = Message sender peer
o = Associated Requested Object
$Index(P)$ = R-SPUN index of message receiver peer P
$type$ = The message type

Output: ‘

$Index(P)$ = R-SPUN index of P after update

1: $\bigtriangledown$ ***Reactive Update based on message type:***
2: **if** $Mes.Type = MISS$ **then**
3: $Index(P).RSRV_N^o[0] \leftarrow Mes.RSRV_N^o[0] - 20$
4: **end if**
5: $\bigtriangledown$ ***Reactive Update based on message*** $RSRV$***:***
6: **if** $Mes.Type = MISS\, or\, HIT$ **then**
7: **for** $i = 1$ to $RSRV_N^o.length$ **do**
8: $Index(P).RSRV_N^o[i] \leftarrow Mes.RSRV_N^o[i]$
9: **end for**
10: **end if**
11: $\bigtriangledown$ ***Reactive Update based on message*** PP_N***:***
12: **for each** $< Object\, o, Success\, s > entry \in PP_N$ **do**
13: **if** s denotes a HIT **then**
14: $Index(P).RSRV_N^o[0] \leftarrow Mes.RSRV_N^o[0] + 10$
15: **else if** s denotes a MISS **then**
16: $Index(P).RSRV_N^o[0] \leftarrow Mes.RSRV_N^o[0] - 20$
17: **else if** s denotes a RSR value **then**
18: $Index(P).RSRV_N^o[0] \leftarrow s$
19: **end if**
20: **end for**

During query forwarding and response returning in the flooding process, every message receiver peer in the query path appends a Bloom Filter representing its local data to the message before forwarding it to the next query path neighbor. As a result a message receiver peer gets a set of Bloom Filters representing documents reachable at various hop distances through a the message sender neighbor. A message receiver peer generates a single Bloom Filter called $unionB$ which contains the $bitwise-OR$ of all the Bloom Filters in the message. This resulting bloom filter has been shown to accurately represent the union of document sets represented by the Bloom Filters in the message [83]. Such a Bloom Filter in a peer P serves two purposes: first, if a object is reachable to a peer within TTL radius but was is not represented in its R-SPUN index, this compensate for the loss of information by allowing a peer to know the object is not rare and is within a reachable distance. Since P maintains a $unionB$ per each neighbor this also allows a peer to identify the subset of neighbors who are in sufficiently close distance to the considered object.

3.3.3 Peer Profile Maintenance

A peer's profile is updated based on First-In-First-Out (FIFO) replacement policy. Peer profiles are exchanged between neighbors by piggybacking with messages sent in the network, without incurring additional overhead. These peer-profile exchanges provides a peer the latest query history of a neighbor which enables it to create local knowledge about neighbor for certain objects it has not queried. This essentially eliminates resorting to the initial random choice of neighbors for a query sent very first time by a peer. Also, this allows peers to dynamically discover how successful the neighbor is at finding objects.

3.4 Rare Object Replication Strategy

Definition 3. *Object Popularity*

We define the object popularity to be the fraction of peers in the network which contains a replica of the considering object. For a P2P network with N peers in an unstructured P2P network, where object o has r copies distributed in r peers , then object o's popularity is

defined as follows:

$$r = \frac{r}{N} \tag{3.3}$$

Due to high number of replicas distributed, finding a popular object in unstructured P2P networks can be performed efficiently. However, when an object is rare, the search scope should be extended to increase the probability of locating the object. Such a solution, however, results is high search traffic. Researchers often times resort to replication of content as the solution. However such methods increase the additional storage and communication cost. Therefore we propose to use Bloom filters for replicating content to reduce space and communication cost. Next we describe our object popularity estimation and replication mechanism.

3.4.1 Object Popularity Estimation

The key issue of replication strategies in unstructured P2P networks is to estimate the popularity of objects distributed in a network. Obtaining such global information in an accurate manner in dynamic, large and unpredictable environments incurs the additional cost of transmitting additional information over the overlay links. We propose to use Spectral Bloom Filters(SBFs) [84] as a space efficient mechanism of measuring object popularity. Spectral Bloom Filters allow associating frequencies of occurrence of individual items in a Bloom Filter. We propose an efficient mechanism to accurately estimate global statistics of object popularity using SBFs as follows: Once peers construct the CDS sub-network, every non-CDS peer periodically randomly selects one of its CDS neighbors to send a Bloom filter containing set of local documents. Upon receipt, the CDS peer aggregates this information to its own Bloom filter representing the same using union equivalent operation. This is bit-wise count addition for Spectral Bloom filters. Then CDS peers communicate with each other to exchange these aggregated Bloom filters. Upon receipt of Bloom filters of other CDS peers, a CDS peer aggregates them to its partially aggregated Bloom filter to complete estimation of global statistics of object popularity. CDS peers then push these Bloom filter containing global object popularity statistics to the previously contacted non-cDS peers.

Using the CDS backbone in estimating global popularity has several advantages. First, it generates much less traffic compared to other techniques such as gossip and is much faster as the popularity estimation communication is localized to a small sub-network, the CDS backbone. The estimation quality is much superior as it is guaranteed that all the peers in the network are contacted in the popularity estimation process. Also, since peers only communicate bloom filter representations of distributed objects, the communication overhead involved is low.

3.4.2 Rare Object Replication

R-SPUN replicates the rare object in the CDS backbone. By distributing the object replicas in the CDS backbone, R-SPUN is able to achieve significant low latency while supporting the successful retrieval of rare objects high probability. Even though the non-CDS peer send "locally rare" objects to be replicated at the CDS backbone, CDS peers take measures to only replicate those objects which are "globaly rare". The basic idea of the scheme is as below.

By comparing the bloom filter containing global popularity statistics against its local data-set, each peer identifies a list of local objects that are rare in the entire network which needs replicating. Once a peer identifies the list of local objects it wants to replicate, it first sends this list to its closest CDS peer. The receiver CDS peer launches a random walk within the CDS backbone to replicate its rare content. A replica of rare objects are established at the peer the random walk terminated. The replication establishment only occurs only for those objects the random walker does not find any replicas at sufficiently closer distances (i.e. within TTL distance) to random walker initializer peer. To determine this, random walker compares the objects to be replicated with the aggregated Bloom filters at each visited CDS peer. This replication strategy improves search efficiency of rare objects effectively. However, this might result in additional communication overhead and storage capacity consumption in replicating objects. To reduce communication and storage overhead, we can simply chose to install a reference to a rare object instead of installing a replica.

We see that our replication algorithm posses many desirable properties. The content replication is performed dynamically without assuming a prior knowledge of object request pattern or peer churn rates. It is a fully distributed, low overhead algorithm.

3.4.3 CDS construction and Maintenance

Constricting a Minimum Connected Dominating Set (MCDS) is a NP-complete problem. There exists many literature focusing on deriving approximation algorithms on constructing Connected Dominating Sets at various costs. In this work we use a simple and distributed approximation algorithm to construct CDS proposed by authrs of [82]. The algorithm uses a marking process that marks every vertex in a given connected and unweighted graph $G = (V, E)$ representing the P2P overlay where nodes represent peers and edges represent links between them. $m(v)$ is a marker for peer $v \in V$, which is either T (marked) or$F$ (unmarked). All vertices are unmarked initially. $N(v) = u|\{v, u\} \in E$ represents the *open neighbor set* of peer v and v has $N(v)$ initially. The marking process is following: (1) Initially assign marker F to every v in V. (2) Every v exchanges its open neighbor set $N(v)$ with all its neighbors. (3) Every v assigns its marker $m(v)$ to T if there exist two unconnected neighbors. The authors show that peer marked as T form a CDS. Authors also propose two rules to reduce the number of peers in the CDS.

Peer Join The authors also puts forward a low overhead CDS maintenance mechanism. When a new peer v joins the network, only v along with its neighbors not in current CDS needs to update their status. The corresponding marking process can be the following: (1) New peer v broadcasts to its neighbors about its joining the network. (2) Each neighbor $w \in N[v]$ exchanges its open neighbor set $N(w)$ with its neighbors. (3) Peer v assigns its marker $m(v)$ to T if there are two unconnected neighbors. (4) Each neighbor $w \in N(v)$ not in current CDS assigns its marker $m(w)$ to T if it has two unconnected neighbors.

Peer Leave When a peer v leaves the network, only CDS neighbors of that leaving neighbor need to update their status, because any non-CDS neighbor will still remain as

non-CDS peer after peer v leaves. The corresponding marking process can be the following: (1) Peer v broadcasts to its neighbors about its leaving. (2) Each CDS neighbor $w \in N(v)$ exchanges its open neighbor set $N(W)$ with its neighbors. (3)Each CDS neighbor w changes its marker $m(w)$ to F if all neighbors are pairwise connected.

3.5 Query Routing

To achieve a high search rate for both popular and rare objects, we propose a search mechanism that takes advantage of the concept of successful query path evolution and the overlay graph property- Connected Dominating Sets- to allow peers to make query routing decisions.

Upon generation or receipt of a query for an object o, a peer P estimates the popularity of the requested object. P's action differs based on whether the object is poular or not.

3.5.1 Popular Object Search

In our work, an object o is popular if at least one copy of the object is within sufficiently reachable distance (bounded by TTL) to P. This is denoted by the either an existence of an entry for o in P's routing index or its bloom filter data structure. If the object has records in its routing index P will compute a score for each its neighbors called a *Path Gradient(PG)* based on the evolution of successful query paths for o through each neighbor.

Best Path Gradient (BPG) Criterion

If the R-SPUN index of P already contains entries for the queried object, the neighbors leading to the most successful query path for object are chosen based on its local knowledge. We define a new metric called Path Gradient (PG) to quantitatively measure the success of a query path of a given neighbor N for a given object o as follows:

$$PG_N = \frac{\sum_{i=1}^{d-1}(RSRV_N^o[i] - RSRV_N^o[i-1])}{d-1} \tag{3.4}$$

where $RSRV_N^o$ is the RSRV of N for o in Ps R-SPUN index and d is the length of the

vector $RSRV_N^o$. One of the major properties of a successful query path is the monotonically increasing property of success ratios assigned to edges between neighbors in the query path. PG metric captures this property by calculating the gradient of success ratios in a query path. A successful query path has a non-negative PG value with large magnitude. For example, if $RSRV_N^o$ = [30,40,50,60] then PG_P=[(40-30)+(50-40)+(60-50)]/3 = 10. PG is calculated by P for each neighbor based on their RSRVs in Ps R-SPUN index for object o. The top k neighbors with highest PG values are then selected to deploy search walkers. However, when the $RSRV_N^o$ vectors of neighbors are of length d=1, only RSR_0 values exist in P's R-SPUN index for neighbors for object o, simply the RSR_0 values for each neighbor is regarded as their corresponding score .

On the other hand if an entry does not exist for o at P's index, P examine each of the bloom filters representing its neighbors to select the subset of neighbors that contain o. Then k neighbors are randomly selected to launch the query. While traditional Bloom Filters do not give an indication of the strength of peers, we may use Spectral Bloom Filters to avoid this situation. Spectral Bloom Filters provide an indication of the frequency of items in the Bloom Filter. The frequency usually represents the number of times the object was inserted into the Bloom Filter and thus can be considered as the number of replicas of o reachable through the neighbor. Therefore considering object frequency of the neighbor as its scores allows us to select the neighbors leading to replica rich areas for o thus increasing probability of finding o successfully.

3.5.2 Rare Object Search

If an entry for the object does not exist in P's index or reachable Objects Bloom Filter, it decides that the object is rare. Then it changes the search mode to CDS routing. Here, the peer selects one of its CDS neighbors and sends the query to the CDS peer. Once in CDS mode CDS peer routes the query along the CDS backbone until TTL expires or a CDS member containing a reference to requested object is found.

Thr search algorithm is stated in Algorithm 3.2.

Algorithm 3.2 *R-SPUN Query Routing*

Input:

Q = Query Message
P = Message receiver peer
N = Message sender peer
o = Requested Object
$SearchMode$ = the search mode
$Index(P)$ = R-SPUN index of P
$B_{Reach}(P)$ = Reachable Documents Bloom Filter of P
$Candidates(Q) = Neighborhood(P)Q.VisitedPeers$
$Neiborhood_{CDS}(P)$ = neighbors of P in CDS backbone
k = the walker count

Output: ‘

R = Retrieved documents and references
$SearchMode$ = the search mode
$Selected(Q)$ = the selected peers for query routing

$Selected(Q) \leftarrow \phi$
$Q.TTL \leftarrow Q.TTL - 1$
$R \leftarrow LocalSearch(o)$
$\bigtriangledown$ ***Found target Object***
if $R \neq \phi$ **then**
    Return a HIT with R and terminate search
else if $P \in CDS\ AND\ P\ has\ a\ reference\ to\ o$ **then**
    $R \leftarrow P's\ reference\ to\ o$
    Return a HIT with R and terminate search
else if $Q.TTL \geq 0$ **then**
    Return a MISS with R and terminate search
else
    if $Index(P)\ contains\ entries\ for\ o$ **then**
        $\bigtriangledown$ ***Regular Search Mode***
        for each Peer $N \in Candidates(Q)$ **do**
            $Score \leftarrow CalculateBPGradient$
        end for
    else if $B_{Reach}(P))\ contains\ entries\ for\ o$ **then**
        for each Peer $N \in Candidates(Q)$ **do**
            $Score \leftarrow Get\ frequency\ of\ o\ for\ N\ in\ B_{Reach}(P)$
        end for
        Sort $Candidates(Q)$ based on scores
        $Selected(Q) \leftarrow$ Top $Candidates(Q)$ with highest score
        $SearchMode \leftarrow Regular$
    else
        $\bigtriangledown$ ***CDS Search Mode***
        $Selected(Q) \leftarrow$ Randomly select 1 neighbor from $CDS(P) \bigcap Candidates(Q)$
        $SearchMode \leftarrow CDS$
    end if
    Return $Selected(Q)$,$SearchMode$ and R
end if

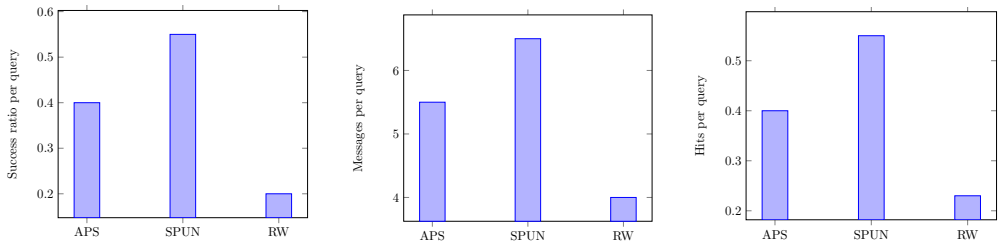
3.6 Experimental Evaluation

3.6.1 Simulation Setup

We simulated the unstructured P2P network using Peersim [85] simulator. The experiments were performed on both random and scale-free network topologies. The object and query replication were chosen from both uniform and zipf distributions. Requesters were randomly chosen and always represented a noticeable fraction (around 10%) of the size of the graph. The default graph had 10,000 nodes with an average out-degree of 10. The default values for walkers and TLL were 12 and 6 respectively. For object replication, we used 100 objects. These objects were selected to be of varying popularity. This was simulated using number of replicas and queries assigned to an object. The highest-ranked 10% of objects amounts to about 50% of the total number of stored objects and receive about half of the requests. The most popular objects were stored in more than 10% of peers while least popular objects were stored in 0.25% of the peers only. The default maximum number of entries per peer profile was set to 10.

3.6.2 Performance Metrics

Three basic performance metrics are used to evaluate the performance of SPUN: (i) success ratio per query, (ii) messages per query and (iii) hits per query. The success ratio is the ratio of successful to total searches made. Success ratio per query is the average success ratio observed for a query. Messages per query is the average number of messages sent for each query. This accounts for the walkers deployed for each query and the return messages. The messages represent the cost of search. Since multiple walkers can be deployed by the query originator, multiple hits can result for a single query, one per each walker. Hits per query is the average number of object replicas found through a query.



(a) (b) (c)

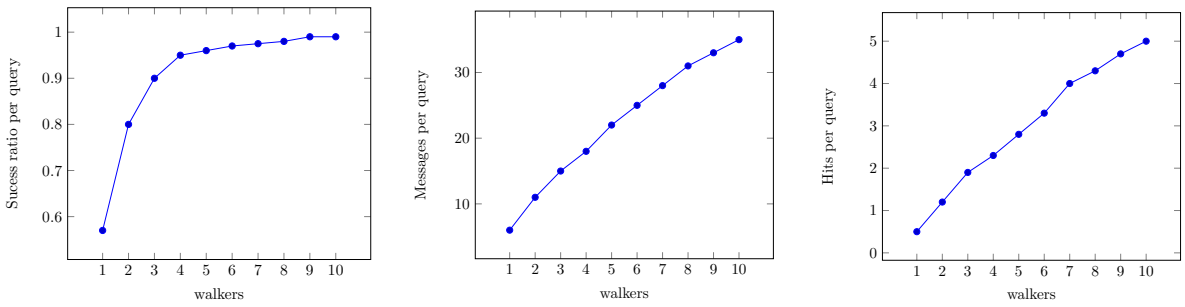
Figure (3.7) Single walker performance. (a) Success ratio per query (b) Messages per query and (c) Hits per query.

3.6.3 Performance Evaluation

We performed extensive simulations to asses the efficacy of the proposed SPUN search algorithm. The performance of the algorithm was compared with two state-of-the-art algorithms, APS [80] and RW [79]. Simulation results obtained with zipf and uniform distributions for query and object replications on random and scale-free topologies were quantitatively similar.

Fig. 5.2 shows the average success ratios, messages and hits per query for each of the algorithms for one walker. The simulation results in Fig. 5.2 demonstrate that SPUN produced the best performance in all three metrics than other algorithms evaluated. As shown in Fig. 3.7(a), SPUN performs 25% better than standard APS in terms of success ratio. The reasons for this behavior are two fold. First, compared to APS which only uses single relative success ratio for peer selection decision, SPUN uses the BPG criterion for neighbor selection that utilize relative success ratios at multiple hops from a querying peer through a given neighbor. Second, SPUN peers have access to more current information about their neighborhoods than APS peers. As shown in Fig. 3.7(b), the high success ratios in SPUN are achieved at similar message complexity of APS. Fig. 3.7(c) shows hits per query for each compared scheme. SPUN showed 20% more hits per query than standard APS when only one walker is deployed.

Fig. 5.3 provides a detailed comparison of the performances between SPUN and APS



(a) (b) (c)

Figure (3.8) (a) Success ratio per query, (b) Messages per query, and (c) Hits per query vs. number of walkers in SPUN.

for varying number of walkers. The objective of SPUN is to achieve high success ratios using few walkers as this reduces communication cost. As Fig. 3.8(a) depicts, SPUN achieves this goal successfully by achieving 25% rate of increase in success ratio than APS. As 3.8(b) shows, SPUN achieves high success ratios with same number of messages produced by APS for any number of walkers. This is because SPUN differs from APS only from the peer selection criteria and content a message carries. Fig. 3.8(c) shows that SPUN outperforms APS in terms of hits returned per query as well. On average, 34.81% of increase in hits per query was observed for SPUN over APS. With increasing number of walkers, we also observed a considerable increase in percentage improvement in hits per query over APS. The effective utilization of more precise results generated by SPUN, results in its performance. Towards the end of plot in Fig. 3.8(a) where higher number of walkers are deployed, the success ratios of APS and SPUN converge towards 100% (shown by 1.0 in Y axis). This is expected when the number of walkers reaches the maximum peer degree when more walkers are deployed (set to 10 in our simulations). In this case, both algorithms default to a limited flooding, where all the neighbors are explored by the querying peer.

Fig. 5.6 compares the average number of hops visited for a search operation by SPUN and other compared algorithms. In both SPUN and APS, a search walker travels 4 hops on average.

Fig. 5.7 shows the relationship between the hits per query and hop distance from the

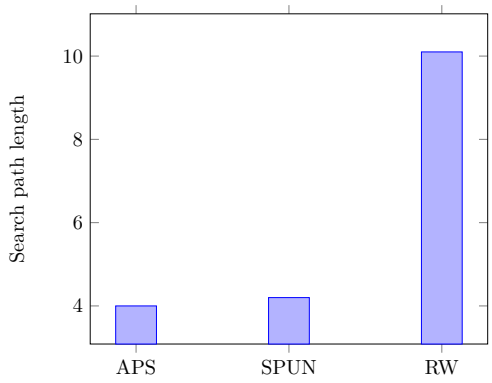


Figure (3.9) Search path length

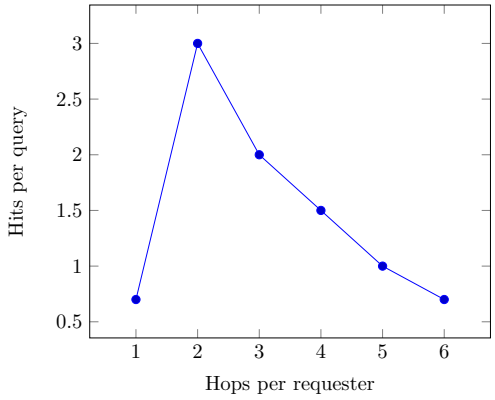


Figure (3.10) Hits per query vs. hop distance from requesters in SPUN

query originator. As can be seen from the figure, SPUN efficiently locates objects at shorter hop distances from the querying peers. Both SPUN and APS show highest hits at 2 hop distance and SPUN shows 20% more hits per query at this hop distance.

The dynamic behavior of the P2P network under peer churn was simulated by adding new nodes to the network, permanently and temporarily removing online peers from the network at varying frequencies. During a simulation, the network size was maintained at roughly 10000 nodes by ensuring the number of nodes joining and re-joining the network were same as that leaving the network temporarily and permanently. Fig. 5.8 shows the success ratios of SPUN and APS at two levels of dynamicity of the P2P network. A simulation run comprised 300 time steps. In a less dynamic setting, the fraction of nodes joining the network, rejoining after temporary removal, leaving permanently, and temporarily removing from the

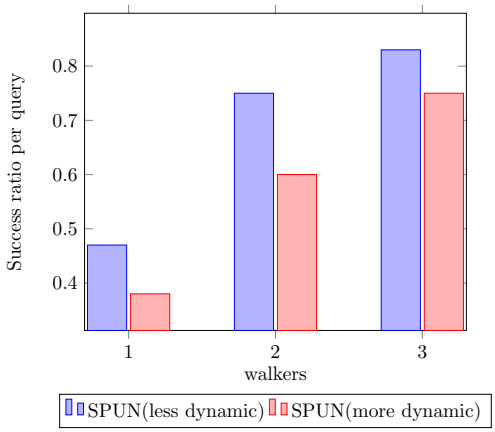


Figure (3.11) Success ratio per query for varying levels of dynamicity.

network were set to 0.2% in each time step. This corresponds to about 20 nodes joining or leaving each time step, or about 80 nodes changing while a walker takes an average of 4 hops. In the more dynamic setting, the same parameters were set to 2% for each time step. Thus, an average walker may be impacted by about 800 node join and leave events. SPUN achieved almost 12% better success ratio compared to APS in the less dynamic setting, and 8% better success ratio compared to APS in the more dynamic setting especially when a single walker is deployed. The reason for the better performance in SPUN in both dynamic settings is that its ability to discover multiple successful neighbors for a queried object using profile exchange mechanism. Therefore, it has a better tolerance than APS, when the network structure changes dynamically.

3.7 Discussion

3.7.1 Message Complexity

In the worst case for a search process, all the k walkers deployed by a querying peer will travel TTL hops and return a hit or a miss along query path. Therefore the number of messages sent per query is $2 \times k \times TTL$ in the worst case. This is the same message complexity of APS. However, unlike APS, a message in SPUN may carry the message sender's profile and a vector of success ratios.

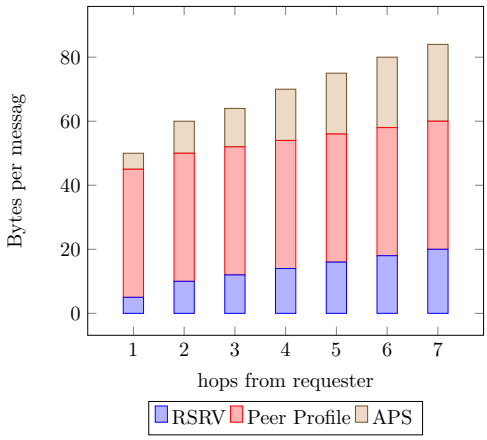


Figure (3.12) Return message size in bytes vs. hop distance from requesters for APS and SPUN.

Fig. 5.9 shows the change in return message size in SPUN with the hop distance from querying peer. In addition to the content an APS return message carry, a SPUN message also carries a peer profile and an RSRV (shown as separate partitions in each bar). Bytes consumed for these three components are also shown in the figure. APS messages contain 15 bytes each. Both algorithms showed a gradual increase in return message size with increasing hop distance from querying peer. This is because return messages carry the visited node list in the query path (shown by APS partition in each bar) which is updated at every visited peer. A return message in SPUN on average traveled 5 hops and average size of a SPUN return message was 71 bytes. This size however is considerably low and does not affect message complexity as the most important parameter - number of messages per query- is same as that of APS. In addition to this, we observed an average SPUN query message size of 47.2 bytes. This is because, sender profile itself consumed 40 bytes.

3.7.2 Space Complexity

Compared to APS, a peer in SPUN maintains a vector of success ratios per neighbor for each object it requested in its local SPUN index. The maximum number of elements in this vector is TTL, as a query can propagate maximum TTL hops only. Therefore, for N such objects and for maximum P neighbors, in worst case, O(PNTTL) space is required by

a SPUN index. APS on the other hand, maintains a single index value per peer per object. Therefore in the worst case scenario, APS needs only $O(P \times N)$ space. In SPUN, each peer also maintains a profile which keeps past queries answered or passed through it. For a maximum number of R records, the profile requires O(R) space. This storage requirement is not a burden to an average internet computing device. However, the peers with low storage capacity can choose to erase some of their locally kept knowledge according to some policy such as FIFO policy or Least Recently Used (LRU) policy.

Our SPUN algorithm eliminates many problems associated with APS. First, it minimizes initial random selection of neighbors in APS using peer profile exchange. Second, unlike APS which has considerable performance degradation in dynamic setting due to its adaptive environment, SPUN is more tolerant in dynamic environments due to dynamic discovery of successful paths using peer profile exchanges. SPUN also has access to more current and larger amount of information than APS. For both dynamic and static networks, SPUN outperforms APS and RW. This performance gain is due to reinforcement learning mechanism a peer exercises which result in local knowledge of a peer getting more precise and less ambiguous over time. SPUN captures the synergistic effect of these more precise success ratios resulting in better selections of peers to send queries. One of the major advantages of our SPUN protocol is that there is no overhead involved in peer arrivals and departures. Also, since RSRs are not related to file content, there is no action required after object updates. Therefore, the maintenance cost is considerably low which is a major advantage compared to many current approaches.

3.8 Summary

SPUN introduces a new peer selection criterion based on the phenomena of development of successful query paths in the P2P network. The best subset of peers selected to send a query are the ones leading to more successful query paths for the queried object. The algorithm is shown to yield considerable increase in success ratio than APS and RW for similar message cost. Our simulations on a variety of environments demonstrated the versatility of

the SPUN algorithm. Results showed that SPUN achieves 25% improvement in success ratio and 20% improvement in hits per query over APS for static networks, and continues to be superior for dynamic networks.

PART 4

ONTOLOGY BASED CLUSTERING

Resource management is very important yet challenging in large scale distributed systems like P2P networks. With more and more users incorporating semantic meta-data with their resources, the resource discovery mechanism should not only be able to scale well with the large number of information resources but also be capable of retrieving semantically relevant resources distributed in the P2P network with high accuracy and efficiency. In this thesis we address these problems by proposing an ontology-based fully-decentralized peer clustering scheme where the network topology is optimized to perform efficient semantic query routing. The proposed semantic clustering scheme utilizes the structural relationships in ontology to organize peers into clusters based on the semantics of the resources they share. Performance evaluation demonstrates that our proposed approach can dramatically improve the search efficiency of unstructured P2P systems while keeping the communication cost at a level comparable with the state-of-art unstructured P2P systems.

Semantic clustering has been studied by the research community as a means of improving search performance of the P2P network. Works such as [17, 22, 41] use Vector Space Model (VSM) based node vectors as the basis for semantic clustering. While it is simple to implement and allows easy clustering of peers based on cosine similarity measurements, VSM is a poor choice for representing semantics. Clustering algorithms that rely solely on statistical correlations may only serve to disrupt the more complex semantic significance attributed to document collections by richer ontologies. SONs [20] is one of the early approaches that used semantic clustering using ontologies but it heavily relies on broadcasting for query routing and peer join. There are several Distributed Hash Table (DHT) based semantic clustering approaches proposed such as [7] to provide efficient exact match searches. Maintenance cost of DHT overlays, however, is costly during frequent joining and leaving of

peers. Moreover, the rigid topology structure may not allow a peer to accept new neighbors or content as peer autonomy is highly restricted in a structured P2P environment.

In contrast to these approaches, we propose a semantic clustering and routing scheme which aims to improve the quality and efficiency of search. The basis for semantic clusters are concepts of an already agreed-upon shared semantic ontology. We use this semantic ontology with the vision of improving information searching and facilitating interoperability. In this thesis we propose a novel dynamic cluster construction technique where peers discover semantic neighbors by taking into account the structural relationships of concepts in the ontology. We also propose a novel query routing mechanism that exploits the concept hierarchy to quickly route a query to the target cluster. Both search traffic and gossiping are employed by peers to acquire global information regarding peer interests.

Our main contributions and results of this work are as follows:

- We develop a novel distributed, dynamic semantic search infrastructure that performs efficient query routing and information retrieval using synergy between ontologies and P2P systems. The proposed ontology-aware topology construction technique exploits global semantic taxonomy relationships between concepts to construct concept clusters and establish connectivity between concept clusters so as to limit the query response time and achieve high recall rates. A peer in a given cluster selectively chooses the concept clusters to establish connections in such a way as to ensure rapid access to target clusters regardless of whether a given concept cluster is semantically closer or further from the concept clusters to which a querying peer belongs.

- We introduce a novel query routing mechanism with concepts like same-cluster-links, family-tree-links, and random-cluster-links where a peer effectively exploits the semantic taxonomical relationships between clusters to quickly route a query to target concept clusters. The hierarchical organization of concept clusters always provides a guaranteed path from one concept cluster to another through parent/child links for query routing.

- We experiment with two techniques for neighbor discovery in cluster construction and maintenance: gossip based neighbor discovery and query traffic based neighbor discovery. In the gossip based neighbor discovery mechanism, a peer simply selects a neighbor at random and exchanges information about the peers it has in its cache. In query traffic based neighbor discovery, a peer uses cache information appended by all peers visited by a query or query response message to discover new peers.

- We compare our algorithm with state-of-the-art search techniques Gnutella [1] and BF-SKIP [41]. The experimental evaluation shows that SAS outperforms the comparison counter-parts for both static and dynamic environments.

4.1 Preliminaries

4.1.1 System Model

We make the following assumptions about our proposed search model:

Network Topology We consider an unstructured P2P network with a set of peers $\{P_1, P_2, ..., P_P\}$with average degree γ. Each peer in the network is able to communicate with its direct neighbors (i.e. one hop neighborhood) only.

Global Semantic Ontology A globally known reference ontology is agreed upon by all peers in the network. This ontology O consists of a set of semantic concepts $C = \{c_1, c_2, ..., c_m\}$ with total m concepts and relationships among them. For simplicity, we assume a semantic hierarchy where the relationships among concepts are only IS-A(i.e. hypernym/hyponym) relations.

Data Distribution There can exist multiple copies of the same documents distributed among peers. For each document a peer has in its local storage, it constructs a *concept weight vector* that depicts the semantic weight of each ontology concept present in that document.For a peer P with a set of documents $D(P) = \{d_j, j = 1, 2, .., n\}$ in its

local storage, we use the following Concepts Vs. Documents matrix $CD(P)$ to describe the concept weights of the documents in P for the concepts in the shared ontology:

$$CD(P) = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \cdots & w_{m,n} \end{bmatrix}$$

where $w_{i,j}$ is the weight of concept s_i where i=1,2,..,m in document d_j.

The weight $w_{i,j}$ of concept c_i in document d_j in peer P is calculated as follows: First, the concept frequencies of all concepts for each document in Ps local storage are calculated by a process of text mining followed by word sense disambiguation. The concept frequencies of each document are added bottom up the semantic hierarchy to account for the contribution from each concept to its ancestor concepts in the hierarchy due to the IS-A relationships. 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 known so far by the peer.

Semantic Query A query consists of the conjunction of one to n concepts in the ontology. The document locating problem therefore is to find as many documents in the P2P network that satisfy the query and that exceed a pre-specified relevance threshold. Existing techniques [51,86] proposed by the research community can be employed to convert keyword queries to concept queries.

Semantic Similarity Calculation We used the following well-known similarity measure *Sim* to calculate the similarity between two concepts c_1 and c_2 in the hierarchy [87]:

$$Sim(c_1, c_2) = \begin{cases} e^{\alpha l} . \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} & if c_1 \neq c_2 \\ 1 & otherwise \end{cases} \tag{4.1}$$

Where l is the shortest path distance between c_1 and c_2 in taxonomy and h is the depth of the least common subsumer of c_1 and c_2, α and β are parameters scaling the contribution of shortest path length l and depth h, respectively where the optimal values were defined as 0.2 and 0.6, respectively [87].

4.2 System Architecture

Here we describe in detail our design of the SAS system that utilizes the synergy between ontologies and P2P systems to provide high quality search performance.

4.2.1 System Overview

Peers in a distributed network are rich in a variety of semantic interests that are based on their sharable local document collections. Our clustering policy is based on the intuition that given a global ontology that describes the semantics of those shared documents, each peer's expertise can be described by a set of concepts in that ontology. This allows one to model different semantic relationships between peers using ontological relationships that exist between concepts and further enables a physical organization of peers into semantic clusters. More specifically, peers specializing in the same concept are physically grouped together as a cluster. A peer can specialize in zero or more concepts based on its shared document collection. This implies that the peer may participate in zero or more concept clusters.

In our SAS framework, we model different types of semantic relationships between peers. A peer P can maintain a *same cluster link* to another peer for common concept they both specialize in. Similarly, P can maintain a *family tree link* to another peer that specializes in a concept that falls under the *family tree* of P. The *family tree* of a peer is the minimal sub-tree in the shared ontology semantic tree that subsumes all the concepts the peer is rich in. For every peer, for every concept for which it has richness of content, a *parent link* and a *child link* is created to the representative parent and child concept clusters. Any concept falling inside its family tree is considered a semantically close concept to its semantic expertise by

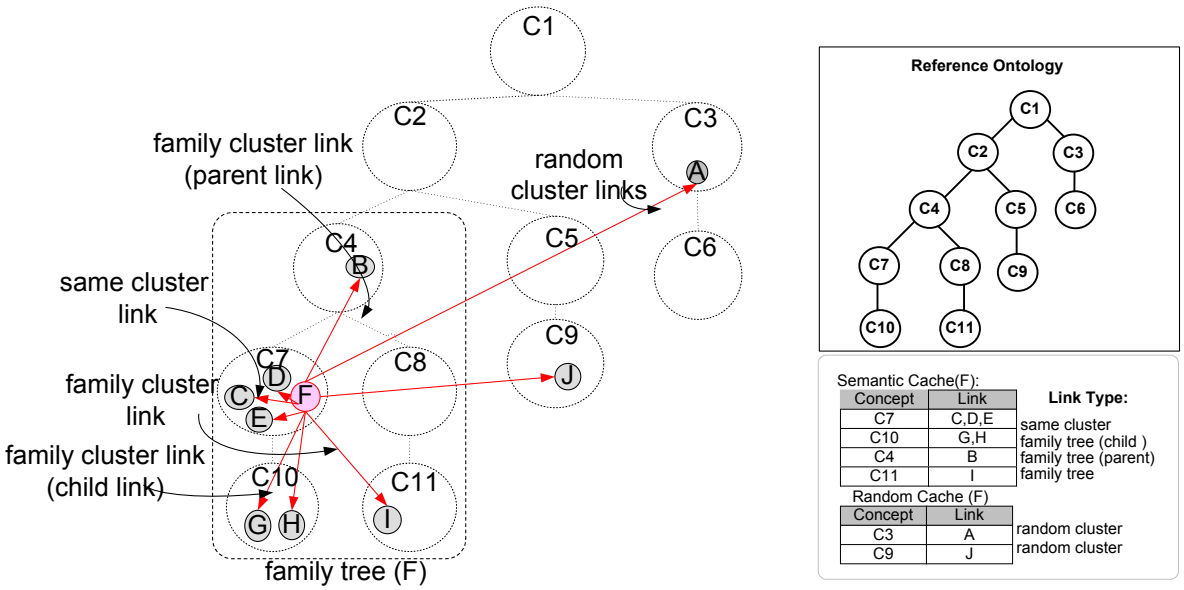


Figure (4.1) SAS Overlay

a given peer. P can also maintain a few random cluster links with some selected peers in the network. These would be peers for which their high frequency concepts fall outside the family tree of P and thus are considered semantically distant neighbors. Therefore, concept clusters self-organize into a hierarchical tree using parent links and child links while also maintaining other types of links for faster accessibility. We propose a scheme where these different types of relationships are effectively exploited to route queries to peers specializing in requested semantics, thus enabling efficient query routing.

Fig.4.1 shows a high level view of the network topology. The dotted circles and lines show the concept clusters and their IS-A relationships in the ontology respectively. While many links are maintained between peers, the different cluster connections of only peer F are shown for clarity assuming a simplified scenario where F belongs only to concept cluster $C7$.

4.2.2 Clustering Policy

Physically grouping peers to create peer clusters based on their semantic richness mandates a good clustering policy. In our work, we use ontology concepts as the basis for peer

clustering. Since peer's interests can be represented by its local data, the ontology metadata is used to extract semantics of shared documents and thus the conceptual interests of the peer. Peers rich in certain concepts in terms of both information content and number of relevant documents for those concepts, categorize themselves as belonging to those concept clusters.

Each document in SAS is represented as a concept vector. These concept vectors contain log weighted concept frequencies *1+log(concept frequency)*. Compared to traditional *tf-idf* measure, log frequency weighting produces higher quality clusters [88] and does not require global information for computation. For a document to be considered relevant for a given concept c_i, the concept frequency of c_i in document should exceed a certain threshold. This threshold concept frequency for concept c_i at peer P is calculated as follows:

$$CFThredh_{i,P} = RelThresh \times MaxCF_i(P) \; where \; c_i \in C(P) \tag{4.2}$$

Where $RelThresh$ is the system specified relevance threshold for a document which is a value in [0-1] range.$MaxCF_i(P)$ is the maximum concept frequency known by P for concept c_i. It is initially computed from Ps local document collection and later it is kept updated by any new higher c_i values discovered from other documents in the network through message exchanges ensuring that eventually reaches its globally equivalent value with time. The calculation ensures P is considered relevant for only those documents having large enough semantic weights for the given concept. This mechanism improves the precision of search over time by allowing a query to identify only those documents that are actually semantically rich and thus relevant for a queried concept. This precision improvement mechanism is briefly described in section 4.3.2 and more details can be found in our previous paper [89].

Once the number of relevant local documents per concept is determined for each peer, there are two basic criteria that can be used to classify a peer to a concept cluster. One way is to conservatively classify a peer to the concept cluster if it merely contains any relevant document for that concept. While this method increases recall, it also increases message

production, peers per cluster as well as clusters per peer. The other method is to classify only highly semantically rich peers (i.e. peers with significantly large number of relevant documents for that concept) for a relevant concept cluster. This has two advantages: first, this reduces the number of peers in a cluster thus reducing query traffic, next it will also reduce number of cluster connections a peer needs to maintain as a peer will belong only to a few clusters. However, this method might result in reduced recall as a query is not able to locate all the relevant documents for the queried concept. On the other hand, while classifying every peer with even a single relevant document for a given concept cluster ensures maximum recall, this will increase the peers per cluster and the message traffic.

4.2.3 Types of Semantic Links

Peers maintain three types of semantic links with their neighbors. To illustrate these links consider Fig.4.1. The figure shows the shared ontology containing concepts $C1$ to $C11$. Let us assume a peer P is rich in concepts $C7$ and $C11$.

Same Cluster links These links are established between peers who belong to the same concept cluster. For example, according to Fig.4.1, peer P can establish same cluster links with other peers in clusters $C7$ and $C11$.

Family Tree links To achieve high semantic reachability, each peer maintains a relatively few number of links called family tree links which are selected among the member concepts in the family tree of the peer. A family tree of a peer P, $familyTree(P)$, is the sub-tree of the ontology rooted at the least common concept that subsumes semantic cluster concepts a peer P belongs to. This least common ancestor (lca) concept of $familyTree(P)$ is called the $lca(P)$. To ensure hierarchical connectivity of concept clusters based on semantic taxonomical relationships, a peer always maintains family tree links for its parent concept cluster and child concept clusters called *parent links* and *child links* respectively. These parent and child links essentially allow the network topology to be organized to mimic the global semantic hierarchy. For the example in Fig.4.1, $lca(P)$ is $C4$ as this is the least common

ancestor that subsumes both $C7$ and $C11$. The $familyTree(P)$ is the sub-tree rooted under $C4$ and is marked in the figure. Any concept in $familyTree(P)$ excluding $C7$ and $C11$ are candidates for establishing family-tree links.

Random Cluster Links To obtain faster access to semantically distant clusters, peers can also maintain a few random cluster links in its random neighbor cache to its semantically distant clusters. According to Fig.4.1, all the concepts in ontology outside family tree of P such as $C5$ and $C6$ are perfect candidates for P to establish random cluster links.

4.2.4 Information Organization at a Peer

A Peer maintains several data structures to aid in clustering and query routing. Each peer in SAS maintains three data structures:

CD(P) $CD(P)$ to describe the concept weights of the documents in P for the concepts in the shared ontology

$MaxVector(P)$ The $MaxVector(P)$ is used for document re-normalizing purposes and holds global statistics of highest information content (calculated as concept frequency) for each concept in the ontology, for entire data set distributed in the network. This information is obtained gradually using message passing mechanisms.

$MyClusters(P)$**)** This is the set of concept clusters in which peer P categorizes itself as a member.

$SemanticNeighborCache(P)$ Semantic neighbor cache keeps track of different concept clusters the peer has discovered and regards as being semantically close to its semantic interests. An entry in the semantic cache takes the form $< c, neighbors >$ where c is the concept cluster for which the peer has established a connection with given neighbors. This includes entries for the same cluster concepts (i.e. concepts in MyClusters(P)) and familyTree

cluster concepts. To avoid unnecessarily adding entries for all candidate familyTree cluster concepts, a newly discovered family tree link is added to Semantic neighbor cache only if the semantic similarity between its representative familyTree concept and the familyTree concept already in cache semantically closest to it is less than a given threshold.

RandomNeighborCache(P) A peer also keeps a second cache called random neighbor cache where it includes pointers to peers, whose clusters are semantically farther away from the clusters the peer belongs to. An entry in the random neighbor cache takes the form $< c, neighbors >$ where c is the concept cluster with which the peer has established a connection with given neighbors. Similar to family tree link addition policy, to minimize the number of entries in random neighbor cache, a newly discovered random link is added to cache only if the semantic similarity between its representative random cluster concept and the random cluster concept already in cache semantically closest to it is less than a given threshold.

Both caches are refreshed using Least Recently Used(LRU) policy.The maximum number of entries allowed in each cache including the maximum number of neighbors allowed per each cluster entry are system specified parameters. Fig 4.1 provides some example cache entries for a peer F with a family tree as shown.

4.3 Dynamic Peer Evolution

4.3.1 Cluster Construction

We assume that the initial network is a pure unstructured network with a randomly connected set of peers. At the configuration stage, each peer constructs document concept vectors for each of its document in the shared document collection. These document concept vectors are used by peers to classify themselves into a set of concept clusters to which they belong.

Neighbor Discovery To establish inter-cluster connectivity, it is crucial that each peer acquire cluster connections by neighbor discovery. We propose two methods to acquire cluster links: (i) a gossip based neighbor discovery which actively discovers new semantic peers and (ii) a passive query traffic based neighbor discovery algorithm.

(i)Gossip Based Neighbor Discovery: Gossip protocols are highly scalable and resilient communication protocols that are widely used to solve problems such as information dissemination, data aggregation etc. when the underlying network structure is inconvenient or extremely large. To implement gossip based neighbor discovery in a P2P network, each peer at each given fixed intervals of time randomly chooses a peer from its neighborhood to exchange information about the concept clusters links they maintain to update their random neighbor cache and semantic neighbor caches with new links learnt. Due to the characteristics of gossip-based algorithms, it is guaranteed that every peer could establish cluster connections with every other concept cluster it is interested in with high probability in logarithmic steps of the size of the network.

Gossiping can be specialized just for the purpose of discovering a disconnected same concept cluster where peer chooses one of its parent or ancestor semantic links instead of just a random link to gossip. In fact, due to the ontology structural relationships, a peer can contact a same-cluster-neighbor to obtain ancestor, descendant, parent and child links and vice versa.

(ii)Query Traffic Based Neighbor Discovery Alternative to gossip based neighbor discovery, we propose a query traffic (i.e. query messages and query response messages) based neighbor discovery method where a peer's local knowledge is propagated along a query path by piggybacking its $MyClusters$ data structure and links maintained in its caches in query traffic messages in a passive fashion. A Peer receiving a query traffic message uses the neighbor links already stored in the message by every visited peer of the message to update its cache. The message receiver peer then appends its local knowledge to the message before forwarding it to the next intended neighbor.Piggybacking information in this way using normal query traffic is an excellent idea to reduce communication cost as this does not incur

additional messages for neighbor discovery.

4.3.2 Cluster Maintenance

Cluster Merging A peer initially starts by creating a concept cluster per each concept it belongs to by adding itself as the only member in the cluster. Understandably, this will create many clusters for the same concepts scattered in the P2P network. Over time, when peers acquire knowledge through gossiping or querying, they will also discover such new clusters for the same concepts they belong to. When this happens, the peer merges its cluster with the newly discovered cluster to form a larger cluster. To achieve this, each peer assigns a unique ID to its concept cluster at the configuration stage generated by a consistent hash function. When a peer merges its cluster with a new cluster, the lower ID of the two is assigned to the merged cluster.

Cluster Re-computing Understandably, data re-computation that occurs as a part of precision improvement mechanism may result in changing the number of relevant documents for certain concept clusters a peer belongs to thereby changing the participation of the peer in certain concept clusters. Therefore, it is necessary to re-compute the clusters a peer belongs to for those concepts for which the data re-normalization occurred. Over time, however, when MaxVector values of a peer P reach their global equivalent values, these updates become unnecessary as the dataset will be normalized against globally stable MaxCFi(P) values. Such a change in a peer cluster in turn mandates the peer to review and update its semantic cluster links.

Precision Improvement Learning Mechanism (MaxVector Learning) The documents distributed in the P2P network are first text processed with the aid of shared ontology and Wordnet thesaurus to arrive at concept vectors per document. These documents are initially normalized per concept using maximum frequency normalization based on the highest concept frequencies *known by each peer* based on its local document collection. However, as long as the actual highest concept frequencies based on entire dataset

distributed in the network is known these normalizations do not produce the most accurate concept vectors. This results in peers wrongfully identifying irrelevant local documents for certain queries as relevant reducing precision of queries. To improve the precision over time, each peer keeps track of most recently known highest concept frequencies in a data structure called MaxVector. New higher concept frequencies are revealed when messages pass through peers (query traffic; no extra messages produced) and MaxVectors are exchanged. This results in peers re-normalizing original concepts vectors of local documents making their semantic representations more accurate. Therefore through these message exchanges, locally known highest concept frequencies reach their actual globally equivalent values, thus increasing query precision over time.

4.3.3 Dynamicity of the System

Peer Joins When a peer joins the network for the first time, it first connects to a random set of peers. The peer first classifies itself to a set of clusters based on its shared document collection and sends a join request to all its neighbors, piggybacking the set of clusters it belongs to. Upon receipt of a peer join request, an existing peer provides its existing semantic and random cache links and the set of clusters it belongs to, to the new peer. The new peer then constructs its initial random cache and semantic cache links by using this information. The highest priority in constructing a peer soft state goes to establishing semantic links, especially same-cluster-links. Therefore, after this initial link establishment, a peer can choose to actively seek same-cluster-neighbors from parent (or ancestor) or child (or descendant) cluster neighbors if it already has none in the cache or vice versa.

Peer Leaves and Failures When leaving the network, a peer provides its semantic and random cache link information to its neighbors along with the notification that it is leaving the network. This information is utilized by a notification receiving peer to update its semantic and random caches with appropriate links, removing soft state maintained for the leaving neighbor and also acquiring appropriate links from the links maintained by the

neighbor leaving the network. If a neighbor crashes, peers eventually discover absence of a peer when query routing to the neighbor fails and it updates its soft state accordingly. Gossip and query traffic based neighbor discovery eventually re-establishes lost links.

4.4 Query routing strategy

The goal of query routing strategy is to locate as many relevant documents as possible. Initially, when no semantic links are acquired, peers resort to flooding the query to its originally connected neighbors. This initial flooding combined with gossip described in section4.3.1 help peers to initially build up their semantic links. Once some links are acquired, peer can intelligently route a semantic query to target semantic clusters avoiding flooding. Examining the queried concepts, a peer first decides which cache it should look up for target cluster. If a direct link is present in cache for the queried concept in the query is directly forwarded to the target cluster. Otherwise it calculates the semantically closest concept in cache and forwards the query to the respective links.

When a query contains a conjunction of multiple concepts, the querying peer constructs a sub-query per queried concept with the same message id with original query included. The basic idea is to propagate the concept based sub-queries to relevant target clusters, where the original query will be evaluated. The query is propagated until a system specified Time to Live (TTL) is reached. The results will return in the reverse path of the query propagation. The query originator then collects and returns the merged results to the end user upon receipt of responses for all sub queries dispatched. Below we describe the inter-cluster routing mechanism that propagates the query to the target cluster and the intra-cluster routing mechanism which takes care of forwarding the sub query to all cluster neighbors with high probability.

4.4.1 Intra-cluster routing

When a peer receives a query targeted at a concept cluster that a peer belongs to, the query enters the intra-cluster routing mode. Here, the peer simply broadcasts the query

through its same cluster links to exhaustively propagate the query within the concept cluster. If the peer does not have any same-cluster-links established yet, it chooses a family-tree-link semantically closest to the target concept cluster, preferably a parent link or a child link to forward the query as those will have the highest probability to maintain a direct link to the target concept cluster.

4.4.2 Inter-cluster routing

If one of the concept clusters the peer belongs to cannot satisfy the query, peers then check if the target concept falls within its family tree. For this to happen, the queried concept should be a member of the family tree of the peer and the peer must have a direct or semantically close family tree link in its semantic cache for the target concept cluster. In this case, query will be forwarded to that concept cluster. If the queried concept falls outside of its family tree, the peer will dispatch the query to the random-cluster-link semantically closest to the queried concept(s). In situations where same-cluster links or family-tree-links are not obtainable, the peer resorts to random-cluster-links to forward a query to a randomly selected cluster.

When the network eventually reaches a steady state, every peer P will establish minimum one parent links and child links per concept in $Myclusters(P)$. Therefore, these parent links and child links provide a guaranteed path to navigate from one concept cluster to another by mimicking IS-A links in the ontology. Therefore a concept query can be propagated from one peer to another until a peer belonging to the target cluster is reached through these parent/child links whichever is semantically closest to the concept query. Therefore, in a steady state network, discovering a link by a new peer will take O(logn) worst case where n is the total number of concepts in ontology. However, the high probability of peers maintaining all other types of links will ensure much faster rate of link establishment of new peer. Additional family tree links and random cluster links allow faster location of the target cluster regardless of its semantic proximity to the peer.

Algorithm4.1 depicts our query routing algorithm. Please note that the background

piggybacking of information through query messages and local state update still occurs when a peer receives a query message. It has been removed from code here to reduce clutter.

4.4.3 Local Search inside a Peer

Once a query is received at a peer, it evaluates the query against its local storage. For a document to be considered relevant for the query, it must contain all the concepts in query and furthermore, should have a relevance exceeding a predefined relevance threshold. For a document d_i in peer Ps local storage, the relevance score of d_i for query Q, $RelevanceScore(d_j, Q)$, is computed as the minimum of its normalized concept weights for those queried concepts:

$$RelevanceScore(d_j, Q) = Mini(w_{ij}) \; where \; c_i \in Q \tag{4.3}$$

where w_{ij} is the weight of concept c_i in document d_j in its concept vector.

4.5 Experiments

4.5.1 Design of Experiments

In this section we present our experimental design parameters such as query generation, document generation, shared ontology, and state-of-the-art algorithms used for comparison purposes.

Ontology The Reuters21578 text classification corpus [90] was used with the category Country as the basis for constructing the semantic taxonomy. We then used the hypernym/hyponym relations of Wordnet ontology to further extend this classification by adding descendant sub trees of each of these core concepts and also to create the core ontology by adding all ancestor concepts of these core concepts in all ancestor concept paths toward the root concept entity in Wordnet ontology. The ontology built this way contains a total of 235 concepts.

Algorithm 4.1 *SAS Query Routing Algorithm*

Input:

Q_{Mes} = Query Message (Q_c = Queried Concept Set, P_v = Visited Peers list, TTL, $MaxVector_{sub}$ = subset of $MaxVector$ of query sender for its most recently updated x concepts, R_Q = Relevant Docs for Q_c per peer in P_v, R_c = relevant docs per concept in C_{Ans} per peer in P_v)
P = this peer executing query routing procedure
Rel_{thresh} = Relevance Threshold
$familyTree(P)$ = the minimal ontology sub-tree containing all concept clusters P belongs to
$myClusters(P)$ = set of clusters P belong to
k_s = same cluster walkers
k_f = family cluster walkers
k_r = random cluster walkers

1: $Q_{Mes}.TTL \leftarrow Q_{Mes}.TTL - 1$
2: $Q_{Mes}.P_v$.add(P)
3: **if** $Q_{Mes}.TTL > 0$ **then**
4: **if** caches $= \emptyset$ **then**
5: flood query to originally overlay neighbors
6: **else**
7: **for each** Concept $c \in Q_c$ **do**
8: Selected(c_i) $\leftarrow \emptyset$
9: $Q'_{Mes}.TTL \leftarrow$ generate Concept Query(c_i)
10: $Q'_{Mes}.TTL \leftarrow Q_{Mes}.TTL$
11: $Q'_{Mes}.R_c \leftarrow Q_{Mes}.R_c$
12: $linkType \leftarrow$ get link type (c_i, $myClusters(P)$)
13: $\triangledown$ ***Same-Cluster routing:***
14: **if** $linkType = Same - cluster - link$ **then**
15: $RelevantDocs(P) \leftarrow$ Local search($Q'_{Mes}.Q_c$)
16: $Q'_{Mes}.R_Q.add(P, RelevantDocs(P))$
17: $Selected(c_i) \leftarrow$get Semantic Cache Neighbors(c_i,k_s)
18: $\triangledown$ ***Family-Cluster routing:***
19: **else if** ($linkType = Family - tree - link$ OR ($linkType = Same - cluster - link$ AND $Selected(Q'_{Mes}) = \emptyset$)) **then**
20: Concept $c' \leftarrow$ get highest similarity concept in semantic neighbor cache
21: $Selected(c_i)$.add(get Semantic Cache Neighbors(c', k_f))
22: **end if**
23: $\triangledown$ ***Random-Cluster routing:***
24: **if** $Selected(c_i) = \emptyset$ **then**
25: Concept $c' \leftarrow$ get highest similarity concept in random neighbor cache
26: $Selected(c_i)$.add(get Random Cache Neighbors(c', k_r))
27: **end if**
28: $\triangledown$ ***send response message:***
29: Send R_{Mes} to Q'_{Mes} sender
30: **end for**
31: **end if**
32: **end if**

Data Generation We used the documents from the Reuters21578 dataset in our simulation experiments. There were 21,578 newswire documents in the dataset and after text processing and words-sense disambiguation, a total of 15,191 documents that produced non-zero length concept frequency vectors were selected. For each of these documents, a concept frequency vector was generated by analyzing the document.

Network Generation We implement our system on top of Peersim simulator [91]. We used the measurement study on real world Gnutella networks [92] as the guideline for network generation and data distribution. We used power law topology to generate the initial network. The default network size was set to 1024 nodes. The initial average peer degree was set to 5. The document distribution among peers follows a Zipf ($\alpha = 1.0$) distribution and each peer contained 100 documents on an average in its local storage. In total we selected 15,191 documents distributed in the P2P network. The TTL varied from 2 to 4 and the default was set to 3. The dynamic behavior was simulated by inserting online nodes to the network while removing active nodes at varying frequencies. On an average, 0.20% of nodes each are added and removed from the network during each simulation time step.

Query Generation We generated 100 random queries each for single and two concept queries from the concepts in the ontology. On average, every peer issues 50 queries during its lifetime randomly selected from generated queries.

In addition to this, the maximum entries allowed per random cluster cache($MaxCache_{Random}$) and semantic cluster cache ($MaxLinks_{Semantic}$)are set to 5 and 100 respectively. The maximum number of neighbors allowed per cluster entry is 5,5 and 10 respectively for random cluster links($MaxLinks_{Random}$), familytree cluster links(Max_{Family}) and same cluster links($MaxLinks_{Same}$). The threshold semantic similarity for adding new family tree link ($SimThresh_{FamilyTree}$) and random cluster link ($SimThresh_{Random}$) are 0.5 and 0.5 respectively.

A combination of gossip and query-traffic based neighbor discovery is used for neighbor discovery mechanism as this proved to give the best performance in preliminary experiments

conducted. The neighbor discovery is initiated every 5th simulation cycle (i.e. after issuance of every 5000 queries). Additional parameters are stated in Table 6.1.

4.5.2 Comparison Algorithms

We compare the performance of our algorithm with the state-of-the-art algorithm BF-SKIP [41] and Gnutella [1] based search.

BF-SKIP Authors of [41] introduce BF-SKIP(Biased Walk, Flooding and Search with K-Iteration Preference), a semantic clustering algorithm based on VSM model. In BF-SKIP, every peer defines a node vector which summarizes the term frequencies of each peer based on its local document collection. A peer maintains two types of connections: random links to semantically irrelevant peers and semantic links to semantically relevant peers(i.e. peers who exceed a given relevance threshold 0.7 based on cosine similarity of node vectors).The neighbor discovery is performed by periodically sending random walk neighbor discovery messages. Given a query, BF-SKIP first relies on biased walks through random links to locate a relevant semantic group, then uses flooding within this group through semantic links to retrieve relevant documents in only one hop. Once in the target cluster, the number of flooded messages in query is iteratively reduced at each hop to reduce unnecessary message production. We set the maximum links to 8 and relevance threshold to 0.7. The iteration depth k is set from 1 to TTL.

Gnutella Gnutella flooding [1] is the most fundamental blind search mechanism where a querying peer floods a query within a TTL hop radius.

4.5.3 Performance Metrics

For our evaluation we rely on following retrieval performance measures:

Recall Recall is the ratio of the number of relevant results(documents) obtained against the total number of relevant results (documents) in the entire P2P network.

Precision Precision is the fraction of documents retrieved that are relevant to a search query.

Message Cost This is the average number of bytes transferred per search query.

Recall per Message This is the ratio of recall to communication cost.

$$Recall\,per\,Query = \frac{Recall}{Communication\,Cost} \tag{4.4}$$

F-Measure We adopted the F Measure to measure the quality of clustering. F Measure is harmonic mean of precision and recall. Each cluster obtained can be considered as the result of a query whereas each pre-classified set of documents can be considered as the desired set of documents towards that query. The precision $P(i,j)$ and recall $R(i,j)$ of each cluster j for each class i is calculated. If n_i is the number of members of the class i, n_j is the number of members of the cluster j,and n_{ij} is the number of members of the class i in the cluster j, then $P(i,j)$ and $R(i,j)$ can be defined as

$$P(i,j) = \frac{n_{ij}}{n_j} \tag{4.5}$$

$$R(i,j) = \frac{n_{ij}}{n_i} \tag{4.6}$$

The corresponding F-measure $F(i,j)$ is defined as

$$F(i,j) = \frac{2 * P(i,j) * R(i,j)}{P(i,j) + R(i,j)} \tag{4.7}$$

Then, the F-measure of the whole clustering result is defined as

$$F - Measure = \sum_{\frac{n}{n_j}}^{i} \underset{i}{max}\left(F(i,j)\right) \tag{4.8}$$

Same-Cluster link establishment rate This measures the percentage of clusters in an average peer's $MyClusters$ data structure that has established at least one same-cluster link.

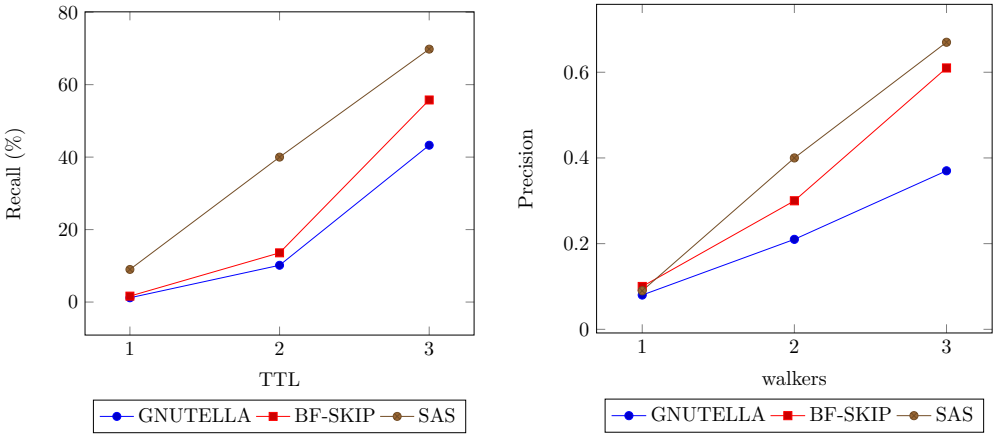
Table (4.1) Simulation Parameters

Parameter	**Range and Default value**
Network size	2^9-2^{15} Default:2^{10}
Ontology Concepts	235
TTL	1-3 Default:3
$RelThresh$	0.7
$MaxCache_{Semantic}$	100
$MaxLinks_{Same}$	10
$MaxLinks_{FamilyTree}$	5
$MaxCache_{Random}$	10
$MaxLinks_{Random}$	5
$SimThresh_{FamilyTree}$	0.5
$SimThresh_{Random}$	0.5
neighbor discovery interval	every 5th cycle

4.5.4 Results and Analysis

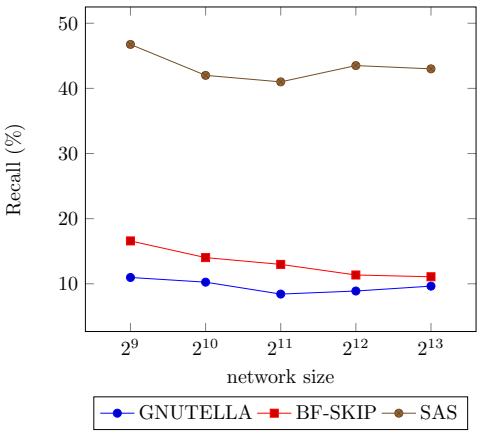
In this section, we discuss the results we obtained for P2P environment. We measured performance of the search algorithms for various network sizes, and results show that performance of SAS is scalable regardless of the network size. Experiments were carried out for both single concept queries and two concept conjunction queries. Due to space limitation, we report only partial results. Results are reported after network convergence based on clustering protocol. Our simulations also show that our SAS algorithm improves recall and precision significantly over BF-SKIP as well as Gnutella flooding counterparts at much lower message cost and produce high quality clusters compared to BF-SKIP.

Query Efficiency Fig. 4.2(a) illustrates the relationship between the query recall and the TTL. The experiments were conducted on a default 1024 nodes network while varying the TTL from 1 to 3. According to Fig. 4.2(a), SAS achieves a high recall rate with small

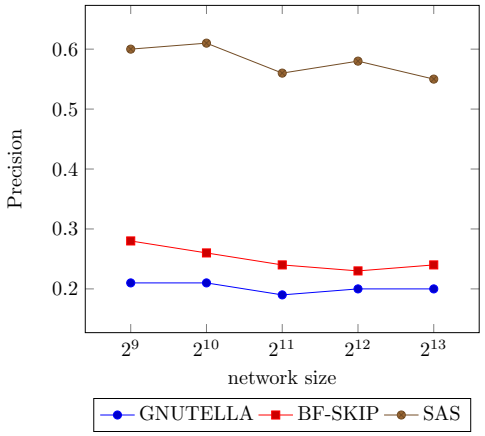


(a) Recall Vs TTL

(b) Precision Vs TTL



(c) Recall Vs Network Size



(d) Precision Vs Network Size

Figure (4.2) Recall and Precision

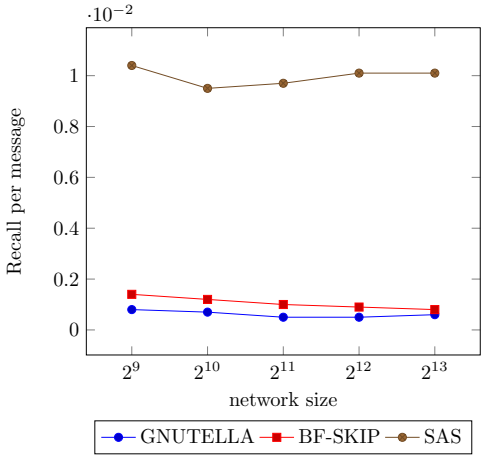


Figure (4.3) Recall per Messages

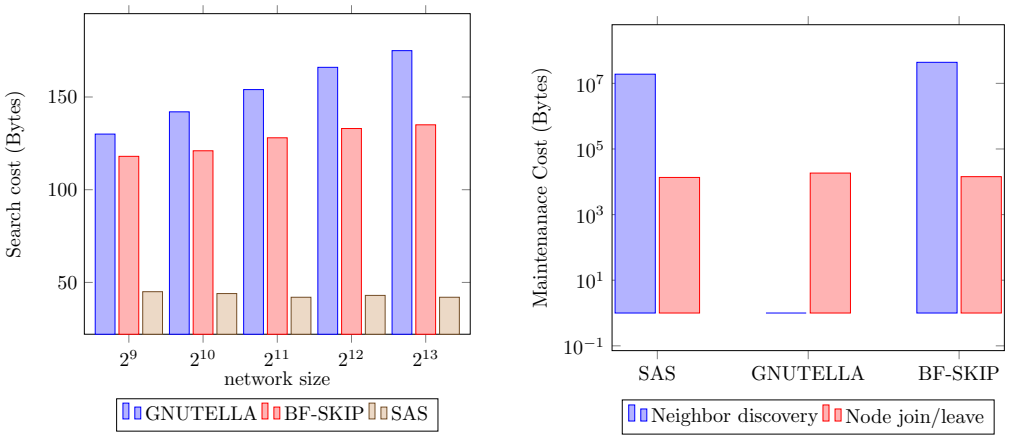
TTL. While our SAS protocol achieves 9.00%, 42.01% and 72.12% recall rates for TTL values 1, 2, and 3, respectively, BF-SKIP achieved only 1.64%, 13.58% and 55.77% recall and Gnutella achieved only 1.18%, 10.15% and 43.28% for the respective TTL values. The reason behind this is that SAS can locate target clusters much faster thereby resolving the query quickly. This indicates that a greater portion of the number of hops travelled by a query is spent within the target cluster flooding semantically relevant nodes thus proving the effectiveness of our link establishment strategy within clusters and between clusters. On average, we observed a 342% improvement in SAS over Gnutella and 223% improvement over BF-SKIP for one concept queries and 350% and 176% improvement over BF-SKIP and Gnutella respectively for two concept queries.

Fig. 4.2(b) illustrates the relationship between precision and TTL. The experiments were conducted on a default 1024 nodes network. According to figure, SAS achieves high precision values compared to BF-SKIP and Gnutella due to the MaxVector learning employed in SAS. Overall we observed 134% and 196% improvement over BF-SKIP and Gnutella respectively for one concept queries and 176% and 122% and 178% improvement over BF-SKIP and Gnutella respectively for two concept queries.

Fig. 4.3 depicts the search efficiency of the three algorithms in terms of recall per message for varying sized networks where TTL is set to 3. For different network scales SAS clearly achieves high efficiency compared to the other two algorithms.

Search and Maintenance Cost Fig. 4.4(a) illustrates the cost of searching in three algorithms.As show in figure, our SAS protocol search cost is significantly lower than that of Gnutella and BF-SKIP regardless of the network size. This is because when given a request, SAS can efficiently locate the target cluster rapidly, so that the search space is reduced and queries get more results with certain TTL. While a 70.23% average improvement of search message cost was observed over BF-SKIP, a 99.44% improvement was observed over Gnutella.

Fig. 4.4(b) depicts the message overhead for a network of size 1024 nodes with search



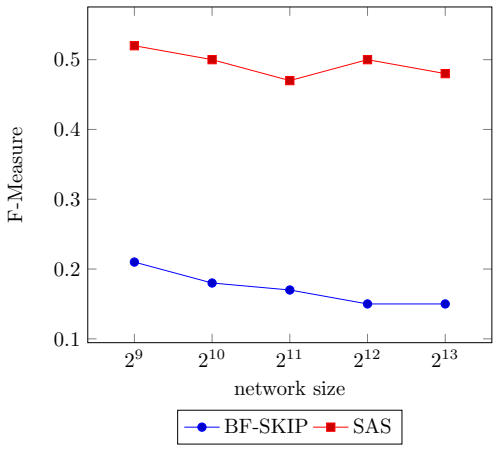
(a) Search Cost Vs Network Size (b) Maintenance Cost

Figure (4.4) Message Overhead

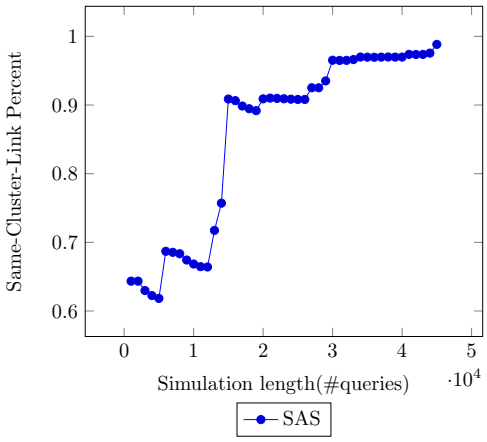
TTL set to 3. Maintenance overhead of all three algorithms accounts for peer join/leave cost. In addition to this both SAS and BF-SKIP include additional overhead of neighbor discovery. The neighbor discovery cost of SAS takes into account the costs incurred in both gossip and piggybacking of information in search messages. Figure reveals that SAS generates the lowest search cost for neighbor discovery queries. Using the combination of search query traffic and gossip messages for neighbor discovery eliminates the need for frequent generation of neighbor discovery queries and thus results in reduced maintenance cost in SAS . The peer join/leave costs are comparably similar in all three algorithms.

Clustering Quality The F Measure analysis is carried out for the two clustering algorithms SAS and BF-SKIP. As shown in Fig. 4.5(a), F Measure values of SAS are much better and consistent regardless of the network size compared to BF-SKIP showing the superior clustering quality of SAS.

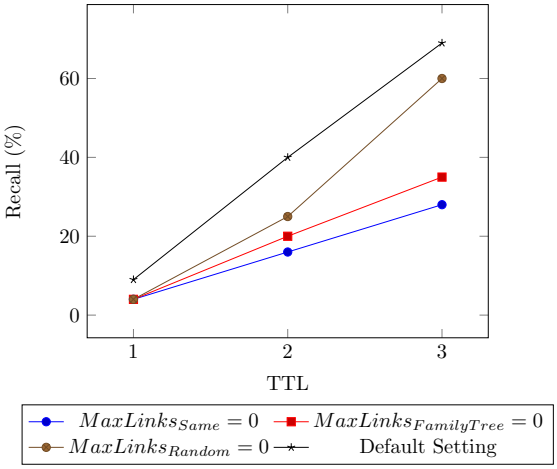
The Fig. 4.5(b) shows the percentage of clusters in its *myClusters*, an average peer establishes at least one same-cluster link with. As can be seen in the figure, regardless of the dynamics of the P2P system, peers gradually establish links with all its concepts in *myClusters* initiating the cluster merging process. This testifies that the neighbor discovery algorithm which is a combination of gossip and search-traffic based neighbor discovery is



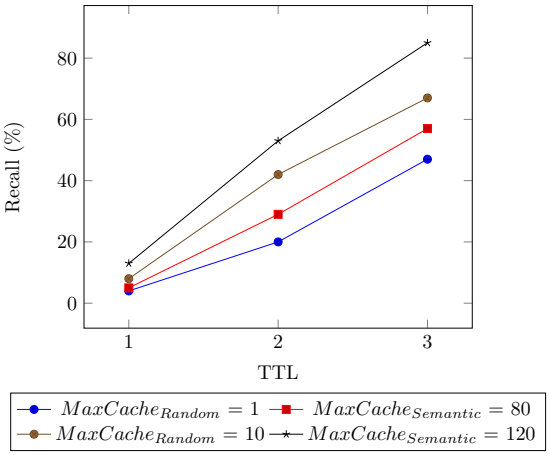
(a) F-Measure



(b) Same-Cluster Link Percent



(c) Effect of Types of Links



(d) Effect of Cache Size

Figure (4.5) Clustering Quality

effective.

Fig. 4.5(c) illustrates the effect of making size of different types of links zero while other parameters assume default values stated in table 6.1. The recall for the default setting as stated in 6.1 is also shown. According to Fig. 4.5(c), it is clear that highest impact on recall is from making $MaxLinks_{Same} = 0$. This impact become more evident in higher TTLs. This behavior is expected as many queries will not be reaching larger portion of relevant peers in the network due to not maintaining same-cluster links. The least impact is made by making $MaxLinks_{Random}$=0. The reason for this is due to establishment of family links and specifically parent/child links, a peer can still navigate a query to the target cluster without aid of random-cluster links.

Fig. 4.5(d) illustrates the effect of cache sizes on system performance. Results show that, increasing the size of both semantic cache and recall increases as both the size of semantic neighbor cache and random neighbor cache increases the recall. The reason behind this is that larger cache sizes allow peers to establish links with more concept clusters increasing the physical distance between any two concept cluster lower.

4.6 Summary

In this thesis we presented an ontology-based clustering and routing protocol that optimizes search performance of the P2P network by ontology-aware topology construction. Our scheme organizes the overlay structure based on semantic concepts and their taxonomical links in the shared ontology. In addition to these taxonomical links, a peer also establishes other types of links to ensure faster location of target clusters and proper dissemination of a query to only relevant peers within a target cluster. While querying, SAS employs a combination of search messages and gossip to discover semantically relevant neighbors.
In the future development of the system, we plan to address more effective dynamic cluster construction techniques, and use of richer ontologies which include relationships beyond simple classification hierarchies. We also intend to experiment with more complex queries and alternate techniques for knowledge discovery.

PART 5

SEMANTIC INDEXING IN UNSTRUCTURED P2P OVERLAYS

5.1 Introduction

Efficient searching for information is an important goal in unstructured peer-to-peer (P2P) networks. While several P2P systems have been proposed for data sharing purposes, many support only semantics-free keyword searches or coarser grained file name searches. In this thesis, we present an ontology based semantic query routing algorithm that performs efficient semantic search in unstructured P2P overlay networks. In our proposed system, the queries are routed in the network by forwarding to peers with highly relevant content in their local storages. To aid in this semantic query routing, we propose a scheme where each peer in the network adheres to a global ontology and semantically tags its local document collection with concepts in the ontology. Based on the semantic tags, peer level semantic summaries are generated, exchanged with neighboring peers and propagated along search paths which aid in efficient local query processing and overlay query routing. An extensive set of simulations performed to evaluate the effectiveness of the system on P2P networks show 380% and 717% improvement in average recall rate, and 410% and 725% improvement in average precision over Ontology Index based Query Routing [20] and Random Walk [18], respectively for dynamic networks at comparable message overheads. Thus, our approach represents a breakthrough in practical terms.

Some of the notable works done in the area of semantic P2P searches include pSearch [25], SemreX [93] and SemSearch [52]. Searching for resources in large distributed systems like P2P networks is a challenging task. With the advent of semantic web, more and more users associate their shared resources with semantic meta-information. This not only allows a user to describe resources from a semantic viewpoint but also lets them specify 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.

Semantic web allows one to associate semantics with World Wide Web resources using a semantic meta-data model. Many current P2P systems incorporate semantic web technologies in their systems with the purpose of providing better knowledge and query representations as well as efficient content locating. A good semantic representation is crucial for local query processing as well as informed query routing. Ontoligies are considered to be the most powerful and expressive semantic knowledge representation model to date. Ontologies are rich data structures that have been successfully used in research as well as in practical applications to represent the knowledge as a set of concepts and relationships between those concepts. Ontologies can be used to represent the semantic concepts, their complex properties, and relationships among the shared documents in a P2P system. Even though there are many well defined semantic technologies such as RDF and OWL freely available to implement ontologies, many fail to represent quantified information or probabilistic information regarding semantic concepts or relations. Also, they require non-user friendly query languages to query.

In this thesis, we present a novel semantic search approach called Ontology based Semantic Query Routing (OSQR) for an unstructured P2P environment. In this scheme, every peer in the network shares a common ontology and each peer is represented by its own semantic index which is generated based on semantic information extracted from its local document collection and from its neighborhood. The objective of the search is to find more relevant documents at shorter distances to the query originating peers with higher recall rates at low message cost. In both local query processing and query routing, we exploit the ontology to provide more semantically relevant results. The summary of our main contributions and results in this thesis is as follows:

- We develop a novel distributed information retrieval search mechanism. A new concept of peer semantic index is introduced for summary representation of a peer,s seman-

tic knowledge. This peer semantic representation effectively combines two important measures to quantify the semantic richness of a peer for a given query: semantic information content of a peer and the total relevant documents reachable through a peer. We incorporate a learning process to incrementally update the peer semantic vectors so the semantic representations improve precision over time when more and more information is acquired by peers about the distributed document collection in P2P system through message exchanges.

- We introduce a novel peer selection algorithm that exploits concept strength in a peer(and a document) and concept relationships in the ontology to provide more semantically relevant results. Compared to a keyword based approach, with little semantic content and laden with a large dimensionality of term vectors, our approach requires limited dimensionality. We also introduce a novel path-based local knowledge updating mechanism that successfully aggregates semantic information gathered through query paths and incorporates into peer's existing soft state.

- We carry out an extensive set of simulations to evaluate our search system. The experimental results show that our OSQR search mechanism is significantly more efficient than the state-of-the-art search mechanisms, Random Walk [79] and Ontology Index based Query Routing [29], in terms of recall and precision at same message complexity. For dynamic networks where the network topology and number of nodes changed over time, OSQR achieved 380.29% and 410.13% improvement over Random Walk for recall rate and precision, respectively, and 717.15% and 725.67% improvement over Ontology Index Based Query Routing in recall and precision, respectively, at comparable message cost.

Compared to existing approaches, our semantic representation model implements a much simpler single dimensional Peer Semantic Index (PSI), that easily allows one to represent semantically quantified information. Also, unlike many previous methods which solely considered simple statistical information such as total relevant documents per concept, our

semantic representation accounts for relevance more accurately by evaluating the semantic content of the document using the ontology, the concept frequency based on subsumed concepts, and by setting a relevance threshold. Thus our knowledge representation is semantically richer and can aid in better query routing and improving precision and recall.

5.2 Prelimineries

We make the following assumptions about our proposed search model:

5.2.1 Network Topology

We consider an unstructured P2P network with a set of peers $\{P_1, P_2, ..., P_P\}$with average degree γ. Each peer in the network is able to communicate with its direct neighbors (i.e. one hop neighborhood) only.

5.2.2 Global Semantic Ontology

A globally known reference ontology is agreed upon by all peers in the network. This ontology O consists of a set of semantic concepts $C = \{c_1, c_2, ..., c_m\}$ with total m concepts and relationships among them. For simplicity, we assume a semantic hierarchy where the relationships among concepts are only IS-A(i.e. hypernym/hyponym) relations.

5.2.3 Data Distribution

There can exist multiple copies of the same documents distributed among peers. For each document a peer has in its local storage, it constructs a *concept weight vector* that depicts the semantic weight of each ontology concept present in that document.For a peer P with a set of documents $D(P) = \{d_j, j = 1, 2, .., n\}$ in its local storage, we use the following Concepts Vs. Documents matrix $CD(P)$ to describe the concept weights of the documents

in P for the concepts in the shared ontology:

$$CD(P) = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \cdots & w_{m,n} \end{bmatrix}$$

where $w_{i,j}$ is the weight of concept s_i where i=1,2,..,m in document d_j.

The weight $w_{i,j}$ of concept c_i in document d_j in peer P is calculated as follows: First, the concept frequencies of all concepts for each document in Ps local storage are calculated by a process of text mining followed by word sense disambiguation. The concept frequencies of each document are added bottom up the semantic hierarchy to account for the contribution from each concept to its ancestor concepts in the hierarchy due to the IS-A relationships. 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 known so far by the peer.

5.2.4 Semantic Query

A query consists of the conjunction of one to n concepts in the ontology. The document locating problem therefore is to find as many documents in the P2P network that satisfy the query and that exceed a pre-specified relevance threshold. Existing techniques [51, 86] proposed by the research community can be employed to convert keyword queries to concept queries.

5.2.5 Semantic Similarity Calculation

We used the following well-known similarity measure *Sim* to calculate the similarity between two concepts c_1 and c_2 in the hierarchy [87]:

$$Sim(c_1, c_2) = \begin{cases} e^{\alpha l} \cdot \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} & if c_1 \neq c_2 \\ 1 & otherwise \end{cases} \tag{5.1}$$

Where l is the shortest path distance between c_1 and c_2 in taxonomy and h is the depth of the least common subsumer of c_1 and c_2, α and β are parameters scaling the contribution of shortest path length l and depth h, respectively where the optimal values were defined as 0.2 and 0.6, respectively [87].

5.3 OSQR Search Protocol

5.3.1 Local Knowledge

A peer P's local knowledge consists of its Concepts Vs.Documents matrix $CD(P)$, its semantic routing index $PSI(P)$, $PSI(N)$ for each neighbor N and finally a vector containing maximum concept frequency known per concept for its local concept set (i.e. set of concepts existing in its local document collection) $MaxVector(P)$. This $MaxVector(P)$ is used for document re-normalizing purposes.

Peer Semantic Index (PSI) : A Peer Semantic Index (PSI) is the semantic knowledge representation of a peer. It shows the goodness of a peer P for semantic concepts contained in its local data storage $C(P)$. We formally represent the peer semantic index of a peer P as a set $PSI(P)$ as follows:

$$PSI(P) = \{(s_i, c_i) \mid s_i \in \Re, c_i \in C(P)\} \tag{5.2}$$

where s_i is the total number of relevant document reachable through P for concept c_i.

Max Vector : The $MaxVector(P)$ in holds the maximum concept frequencies of each concept over all documents distributed in the P2P network known so far by P and therefore represented as a set $\{w_i \mid c_i \in C\}$. The main objective of this data structure is for P to acquire knowledge of richness of documents distributed in the network in terms of their information content, thereby minimizing reporting local documents with low information content compared to those that exist in the network, in response to a query. This essentially improve the precision of the search mechanism.

5.3.2 Ontology based Semantic Search

Peer Selection OSQR is in nature a controlled flooding algorithm where query is intended to find documents that contain all of the concepts specified in a query exceeding a given relevance threshold. A search process is initiated when a peer originates a query by deploying multiple search walkers which is a system specified parameter. Each walker can follow its own path in the network. Upon receipt of a query, a peer searches for matching documents in its local storage by ranking its documents based on a relevance score. The documents that exceed a given threshold of relevance are selected as matching documents for the query. The query is also further propagated to more promising neighbors. A walker propagates TTL hops only. Once all the walkers for the query terminate, the results are returned to the query originator along the reverse path of the query back to the query originator.

In OSQR, a peer forwards queries to the most promising set of neighbors. The most promising neighbors are selected based on the neighbor PSIs a peer maintains of its immediate neighborhood. Peer P calculates a relevance score for each neighbor N for the given query Q based on the PSI of that neighbor. For a neighbor N with $PSI(N)$ at P, the relevance score of N for query $Q = \{c_i \mid c_i \in C\}$ can be formally defined as follows:

$$Relevance(N, Q) = MIN(s_i)\, where\ < s_i, c_i > \in PSI(N)\, at\, P\, and s_i \in Q \qquad (5.3)$$

The relevance of neighbor N for a query is computed as the minimum of semantic weights

for queried concepts. This is because the query is a conjunction of concepts and minimum is the conservative outcome (i.e. relevant documents) from N . Once the relevance score of each neighbor is calculated, the neighbors are ranked in descending order of their relevance scores and top k neighbors are selected to deploy the query. The query originator peer selects a predefined number of most promising neighbors to send the query while the other peers receiving the query merely process and forward it to only one promising peer if the message TTL has not exceeded.

The search algorithm is given in Algorithm 5.1.

Local Search Once a query is received at a peer, it evaluates the query against its local storage. For a document to be considered relevant for the query, it must contain all the concepts in query and furthermore, should have a relevance exceeding a predefined relevance threshold. For a document d_i in peer P's local storage, the relevance score of d_i for query Q, $RelevanceScore(d_i, Q)$, is computed as the minimum of its normalized concept weights for those queried concepts:

$$Relevance\,Score(d_i, Q) = MIN_i(w_{ij})\; where\; c_j \in Q \tag{5.4}$$

5.4 Construction and Maintenance of Peer Semantic Index

5.4.1 PSI Construction

At the configuration stage when a peer joins the network for the first time, it initializes each of its semantic weights s_i per each concept c_i in its PSI with total relevant documents in its local storage for that concept. Once connected to the network and after the construction of its own PSI, each peer P exchanges its PSI with each of its direct neighbor N (i.e. one hop neighborhood). These collected PSIs from a peer's neighbors essentially serve as the local knowledge the peer has about its neighborhood that is exploited by the peer in forwarding queries to most promising neighbors. These neighbor PSIs received are further used to update peer's own PSI, so that each semantic weight s_i per concept c_i reflects the total reachable

Algorithm 5.1 *OSQR Query Routing*

Input:

P = Peer executing query routing procedure
Q = Query Message (Q_c = Queried Concept Set, P_v = Visited Peers list, TTL, $MaxVector_{sub}$ = subset of MaxVector of query sender for its most recently updated x concepts, R_Q = RelevantDocs for Q_c per peer in P_v, R_c = relevant docs per concept in C_{Ans} per peer in P_v)
k = Walker count
x = concept count in $MaxVector(P)$
C_{Ans} = concepts including $Q.Q_c$ and their ancestor concepts in ontology
Candidates(Q) = Neighborhood(P) $Q.P_v$
NeighborSematics= $PSI(P_i) \mid P_i \in Neighborhood(P)$

1: $Selected(Q) \leftarrow \phi$
2: $Q.TTL \leftarrow Q.TTL - 1$
3: $Q.P_v$.add(P)
4: $RelevantDocs(P)$ = Perform local search($Q.Q_c$)
5: $Q.R_Q$.add(P,RelevantDocs(P))
6: **if** $Q.TTL \geq 0$ **then**
7: $\triangledown$ ***Update local knowledge at P***
8: Perform MaxVector Updating, Data Renormalization and PSI Update using Q
9: $\triangledown$ ***Append local information to message***
10: $Q.MaxVector_{sub} \leftarrow$ most recently updated x concepts in MaxVector(P)
11: **for each** Concept $c \in C_{Ans}$ **do**
12: $Q.R_c$.add(P, c, Relevant documents for c in P)
13: **end for**
14: $\triangledown$ ***Calculate relevance strength of neighbors for query***
15: **for each** Peer $N \in CandidateNeighborhood(Q)$ **do**
16: $Score \leftarrow CalculateScoreusingequation5.3$
17: **end for**
18: $\triangledown$ ***Send query to top k neighbors***
19: Sorted $\leftarrow$ Sort neighbors in descending order of relevance
20: Selected(Q) $\leftarrow$ get Top k Peers(Sorted)
21: Forward Q to Selected(Q)
22: **else**
23: $\triangledown$ ***Generate response message*** $R \leftarrow$ generate response message
24: $\triangledown$ ***Append local information to message***
25: $R.MaxVector_{sub} \leftarrow$ most recently updated x concepts in MaxVector(P)
26: $\triangledown$ ***Send response message***
27: Send R to Q.sender
28: **end if**

relevant documents per concept through P. The weight s_i associated with a concept c_i in a PSI in peer P is calculated as follows:

$$s_i = N_{i,P} + Avg\left(\sum_r \frac{N_{i.P_r \in QueryPath(N)}}{\alpha(P, P_r)}\right) \tag{5.5}$$

Where, $N_{i,P}$ is the total number of relevant documents for concept c_i in P's local storage, P_r a peer reachable to P through neighbor N from a query path (or query response path) QueryPath(N) . N_{i,P_r} , is the total relevant documents for concept c_i in peer r P's local storage and a distance based weighing factor. (,) r P P ensures that the farther away a peer r P is from P, higher the distance based weight applied in i s calculation. We use the hop distance in number of hops from P to r P as the current $\alpha(P, P_r)$. Thus $Avg\left(\sum_r \frac{N_{i.P_r \in QueryPath(N)}}{\alpha(P,P_r)}\right)$ represents the average number of relevant documents reachable for i c per hop from P through neighbor N . This is the aggregate reachability of documents for concept c_i from P through a neighbor N .The QueryPath(N) is the best possible path known by P so far that goes through P from a neighbor N and gives highest number of relevant documents per hop distance for the considered concept. An existing s_i value is replaced by a newly calculated s_i , only if this new value exceeds the current s_i.

For example, consider that for a concept c_i , current s_i value in a peer P is 5, and 2 comes from its local document collection(i.e. $N_{i,P}$). Assume P receives a query along the path $P_A \rightarrow P_B \rightarrow P_C \rightarrow P$ through a neighbor P_C. Also assume that for concept c_i, P_A contains 9 documents in its local storage (i.e. N_{i,P_A} , =9), P_B contains 8 documents in its local storage, and P_C contains 5 documents in its local storage and this information is included in query received at P. Now P calculates a new $si = 2 + Avg(9/3 + 8/2 + 5/1) = 2 + 4 = 6$. Therefore previous s_i will be replaced by 6 in P's PSI. As mentioned before, for a document to be considered relevant for a given concept c_i, it should contain a concept frequency exceeding a certain concept frequency threshold for c_i. This concept frequency $CFThresh_{i,P}$, for concept c_i at peer P is calculated as follows:

$$CFThresh_{i,P} = RelThresh \times MaxVector_i(P)\, where\, c_i \in C(P) \tag{5.6}$$

Where $RelThresh$ is the system specified relevance threshold for a document which is a value in [0-1] range. MaxVector(P) i is the maximum concept frequency known by P for concept c_i. This is the maximum concept frequency for concept c_i that P discovers from its local documents and through messages it received from its neighbors in the past. This threshold calculation ensures P is represented by only those documents having large enough semantic weights for the given concept. For example, let the known maximum number of occurrences of concept c_i in a single document be 40, If the threshold is set to 0.7, those documents that have $40 \times 0.7 = 28$ or more occurrences of c_i will be considered relevant.

5.4.2 PSI Maintenance

Each time a peer sends or forwards a query or a query response to one of its neighbors, it piggybacks its total relevant documents for most recently updated concepts in its PSI and $MaxVector(P)_i$ values for intelligently selected set of concepts to its message recipient. As a result, the following three updating mechanisms (i.e. (i)MaxVector learning (ii) data renormalization and (iii)PSI Updating) occur in a peer for improving the semantics of a peer's document collection during query forwarding (forward update) as well as in return message sending (backward update). The ultimate goal of these updating mechanisms is to achieve highly precision in query execution. Over time, peers' MaxVector values will reach their global equivalents with high accuracy resulting in the its normalized data collection being highly precise thus leading to high precision in local query evaluation.

(i) MaxVector learning We incorporate a learning process with MaxVector of a peer P. MaxVector(P) in P ideally should represent the maximum concept frequencies of each concept over all documents distributed in the P2P network. Since acquiring such global information is not feasible in a completely decentralized network, each peer P initializes its MaxVector(P) with local maxima of concept weights per each concept ci calculated from its local document collection. These are later replaced by new concept weights whenever new concept weights that exceed peers current maxima are discovered by that peer during the

query process.

MaxVector learning is achieved with the aid of normal query traffic. Whenever a peer receives query or a query response from one of its neighbors, it uses the neighbor's subset of MaxVector appended to the message to update its own MaxVector. Then it forwards the query (response) to the next message recipient by replacing the subset of MaxVector in the received message by its most recently updated set of concepts in its own MaxVector. For a message sending peer P, the subset of its MaxVector in message is a set of key-value pairs that takes the form $< c_i, MaxVector_i(P) >$. Updating MaxVector(P) at a given peer P requires some computation, as the peer and its neighbor N have different local concept sets. However, these concepts may still be related by hypernym/hyponym relations in the semantic ontology. Due to these hypernym/hyponym relations, for each$< ci, MaxVector_i(N) >$ key-values pair received from a neighbor N ,MaxVector(P) values for concept c_i and its ancestor concepts existing in P's MaxVector need to be updated. Over time, these MaxVector(P) values will propagate over the network through query messages and query responses, and thus each will eventually approach its global maximum equivalent. Algorithm 5.2 states the $MaxVector_i(P)$ updating process in a given peer.

Algorithm 5.2 *MaxVector Updating*

Input:

P = Peer executing the update
N = Neighbor who provided information for updating P's MaxVector
MaxVector(N) = ¡concept, Max(w_{ij})¿key-value pair for last updated n concepts of neighbor
MaxVector(P) = ¡concept,Max(w_{ij})¿key-value pair for concepts of this peer's local concept set
C(P) = Local concept set of P based on its local documents

1: **for each** Concept $c \in MaxVector(N)$ **do**
2: AncestorConcepts.add(ancestor concepts of c in P's local document collection)
3: **end for**
4: **for each** Concept $c \in AncestorConcepts$ **do**
5: **if** MaxVector(P).w_i ¡ MaxVector(N).w_i **then**
6: MaxVector(P).$w_i \leftarrow$ MaxVector(N).w_i
7: **end if**
8: **end for**

Note however, that the number of updates made to a MaxVector(P) at a peer P should

ideally diminish over time as maximum concept weights for those concepts maintained locally reach their global maxima equivalent from previous updates. Therefore, subsequent messages exchanged between peers should not carry maximum concept weights for unnecessarily large number of concepts selected blindly. Hence, to reduce the time required for updates, each peer intelligently selects a subset of its MaxVector values to be piggybacked to its neighbor in the following manner:

Assume a scenario where a peer P has two neighbors A and B. P receives a query from A which in turn it sends to B. Upon receipt of the corresponding query response from B, P forwards it to A. Each peer in the system keeps track of its set of most recently updated n concepts in its Maxvector and piggybacks some entries in query messages it sends to its neighbors. Therefore when P receives a set of $< c_i, MaxVector_i(A) >$ key-values pairs from A through a query, it updates only those concepts in its MaxVector that has a significantly lower maximum concept frequency than that is stated in the received $< c_i, MaxVector_i(A) >$key-values pairs (set to 50% in experiments). For example, assume an oversimplified scenario, where A sends P in the query message, $\{(c_1, 150), (c_2, 90)\}$ as the set of $< c_i, MaxVector_i(A)$ ¿ key-values pairs for the most recently updated two concepts in its MaxVector(A), P has $\{(c_1, 30), (c_2, 80)(c_3, 50)\}$ in its MaxVector(P), and in ontology structure, c_3 happens to be an ancestor concept of c_2. Given that the update threshold is 50%, P will therefore update its MaxVector to $\{(c_1, 150), (c_2, 80), (c_3, 150)\}$ as concepts c_1 and c_3 have lower than $150 \times 0.50 = 75$ maximum concept frequency recorded in P's MaxVector. Conept c_2 however already has greater than $90 \times 0.50 = 45$ maximum concept frequency recorded in P's MaxVector and therefore will not be updated. P also temporarily keeps a copy of this received$< c_i, MaxVector_i(A) >$ key-value pairs from the query sending neighbor until it receives a query response from B. Upon receipt of query response from B, P updates its own MaxVector(P) with $< c_i, MaxVector_i(B) >$key-values pairs embedded in the query response and then compares its just updated MaxVector(P) with temporarily stored$< c_i, MaxVector_i(A) >$ key-value pairs sent from query sending neighbor A to select only those concepts in its local concept set that have a drastic increase over maximum

concept frequency stated in relevant concept in received $< c_i, MaxVector_i(A) >$ key-value pairs (again set to 50% in experiments) to be piggybacked in the query response forwarded to its neighbor A.

(ii) Data Renormalization The updating of $MaxVector_i(P)$ values for certain concepts of a peer P in turn triggers re-computing of normalized concept weight vectors for its local documents where these concepts exist. Therefore over time, the normalized weights in DC(P) of P reach the more accurate globally normalized values. This helps in retrieving more accurate results at local query processing reducing the opportunity to incorrectly identify irrelevant documents as relevant to a query. To reduce the computation cost, this process can be done for only those concepts updated in P's MaxVector and only when a significant change in MaxVector values for those concepts are detected.

(iii) PSI Updating Updating of a peer P's own PSI is triggered by two events (a) receipt of a set of $< c_i, MaxVector_i(B) >$ key-values pairs of and (b) receipt of $< c_i, N_{i,B} >$ key-values pairs per hop distance. The c_i values in $< c_i, N_{i,B} >$ include not only queried concepts, but also the ancestor concepts of those queried concepts in the ontology. Thus, the updating of PSI using (b) allows a peer to gather goodness of neighbors for concepts in the ontology prior to querying for the same and thus this greatly reduces the initial random choice of neighbors in query forwarding. While (a) affects the former part (i.e. $N_{i,P}$, the locally relevant documents per concept) of (1) , (b) effects the latter part of (1)(i.e. $Avg\left(\sum_r \frac{N_{i.P_r \in QueryPath(N)}}{\alpha(P,P_r)}\right)$,the aggregate relevant documents reachable per concept through the best known query path so far).Each time a peer receives a query or a query response, it utilizes the information embedded in the message to update its PSI. Before sending, each peer appends its ID , the total number of relevant documents for queried concepts and their ancestor concepts in its local storage to the message. Therefore, while a $< c_i, MaxVector_i(B) >$ key-values pairs propagate only one hop distance, $< c_i, N_{i,B} >$ key-values of each peer in query path are embedded in the forwarded message propagated along query (or query response) path. These are utilized for PSI updating by each message receiver

peer. In (a), receipt of $< c_i, MaxVector_i(B) >$ key-values pairs will result in receiver peer P updating its necessary entries in its MaxVector. This in turn leads to data re-normalization in P which in turn leads to recalculating of its PSI, as the total local relevant documents for those concepts whose $MaxVector_i(P)$ values have been just updated may have changed. In (b), the query path information is available to a receiver to calculate the latter part of (1): $Avg\left(\sum_r \frac{N_{i.P_r \in QueryPath(N)}}{\alpha(P,P_r)}\right)$ and recalculate (1). An s_i value recalculated this way replaces current s_i value in PSI only if this newly calculated value exceeds current value. Also, when a new set of $PSI_i(A)$ values are received from a neighbor A, the appropriate entries in PSI maintained for neighbor A in P's soft state are replaced by these most recent values.

When a peer leaves the network, it informs its direct neighborhood that it is leaving the network by sending a message. This results in the neighbors removing the leaving peers PSI from their soft state and updating their MaxVector accordingly. When a peer joins the network, it goes through the same initialization process given in step 1 to construct its PSI and exchange the PSIs among neighbors. When changes in peer contents occur, the peer updates its PSI and notifies its neighbors who will, as a result, update the current PSI maintained for neighbor by the new information provided in the notification.

5.5 Design of Experiments

In this section we present our experimental design parameters such as query generation, document generation,shared ontology, and state-of-the-art algorithms used for comparison purposes.

5.5.1 Semantic Ontology

We used the Reuters21578 text classification [90] corpus of newswire documents as the basis of our semantic ontology. The Reuters ontology text classification contained four core concepts into which the documents were categorized: Organization, Exchange, Person and Country. We used hypernym/hyponym relations of Wordnet ontology to further extend this classification by adding descendent sub trees of each of these core concepts and also

to create the core ontology by adding all ancestor concepts of these core concepts in all ancestor concept paths toward the root concept entity in Wordnet ontology. The ontology built this way contains a total of 11,813 concepts. A partial view of the used Reuters semantic hierarchy is given in Fig. 5.1.

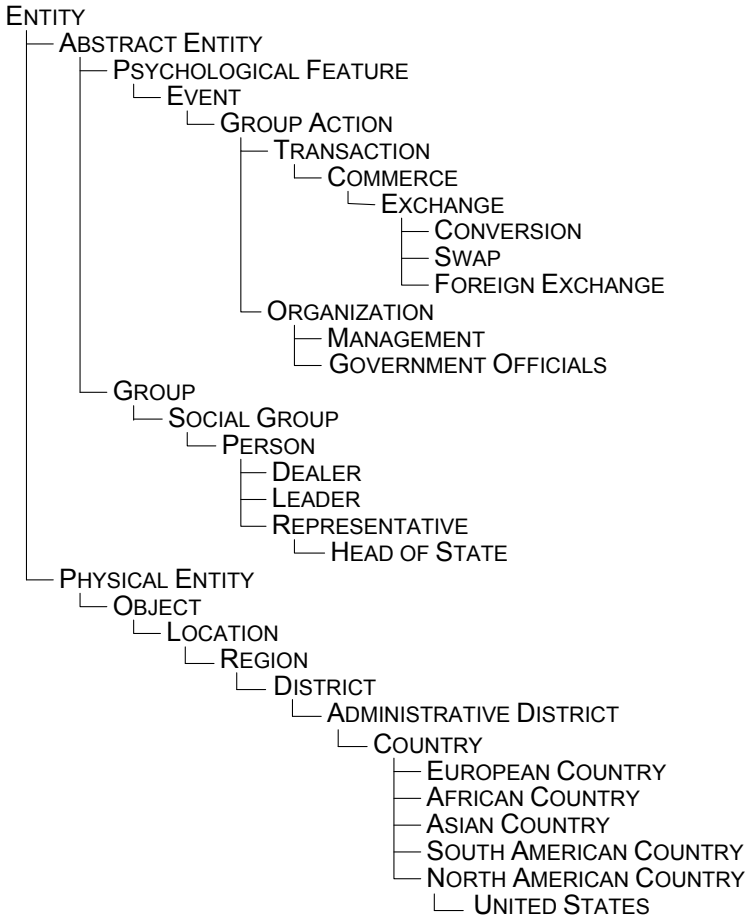


Figure (5.1) Extended Reuters21578 Core Semantic Hierarchy

5.5.2 Data Generation

We used the actual documents from the Reuters21578 dataset in our simulation experiments. There were 21,578 newswire documents in dataset and after text processing and words-sense disambiguation, a total of 15,191 documents that produced non-zero length *concept frequency vectors* were selected. For each of these documents, a concept frequency vector was generated by analyzing the document. Then these concept frequencies per document

were updated by recursively adding the descendants' concept frequencies. The documents were replicated according to a Zipf distribution into multiple copies for allocating to peers.

5.5.3 Network Generation

We implement our system on top of Peersim [91] simulator which is a discrete-event time-stepped simulator for P2P systems. The simulation was carried out over networks generated with Gnutella scheme following the power law topology. The network size was varied from 100 to 100000 nodes Average peer degree was set to 10. The common ontology used is the extended Reuters21578 semantic hierarchy [90]. The document distribution among peers follows a Zipf (α=1.0) distribution and each peer contained 100 documents on an average in its local storage. In total we selected 15,191 documents distributed in the P2P network. The queries were the network (α=1.2). The default TTL was set to 7 to limit the number of hops of walkers. Percentage increment threshold for MaxVector update is set to 50% to ensure that recalculation of document-wide concept frequency vectors is infrequent. 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. On an average, 80 nodes each are added and removed from the network during each simulation run.

5.5.4 Query Generation

We generated 100 random queries as conjunctive queries from the concepts in the ontology. The root concept Entity was excluded from the candidate set of concepts to generate a query. The number of concepts in a query was varied from 1 to 2. Only meaningful queries were generated. A query is defined to be meaningful if no two concepts in the query are involved in an ancestor-descendent relationship.

5.5.5 Comparison Algorithms

We compare the performance of our algorithm with the state-of-the-art algorithm Ontology based Index for Unstructured Networks [20] and Random Walk [18] based search.

They work as follows:

1) *Ontology Index Based Query Routing (OIQR)*: Authors of [29] introduce constant size ontology-based indexing approach for unstructured P2P networks.A matching function which uses the number of documents accessible via a link to rank and select neighbors for query forwarding. OIQR assumes a global ontology. For a fair comparison, to obtain best-case performance of OIQR, we assumed that the predefined index size is equal to the total number of concepts in the ontology. We set its terminating condition to be TTL.

2) *Random-Walk-based Query Routing (RWQR)*: Random walk [79] is a popular blind search mechanism where a querying peer deploys k search walkers by sending query to k random neighbors. The peer selection mechanism does not require a peer to maintain any local knowledge about its neighborhood in this method.

5.5.6 Performace Metrics

For our evaluation we rely on five key measures:

- *Recall*:Recall is the ratio of the number of relevant results obtained against the total number of relevant results in the entire P2P network.

- *Precision*:Precision is the fraction of documents retrieved that are relevant to a search query.

- *F-Score*:This is the harmonic mean of precision and recall. This provides an overall measurement of system efficiency by considering both recall and precision. We use F1 score as F measure:

$$F1\,Score = 2.\frac{Precision \times Recall}{Precision + Recall} \tag{5.7}$$

- *Message Cost*:This is the average number of messages generated per search query.

- *Hits Per Query*:This is the average number of distinct relevant documents discovered per search query.

5.6 Results and Analysis

In this section, we discuss the results we obtained for P2P environment. Figures 5.2 through 5.5 summarize the results. We measured performance of the search algorithms for various network sizes, and results were consistent regardless of the network size. Therefore, we present the results for network of 1000 nodes only. Our simulations show that our OSQR algorithm improves recall and precision significantly over Random Walk (RWQR) as well as Ontology Index based search (OIQR) counterparts at a comparable message cost while keeping PSI update cost low.

5.6.1 Recall

We measured the recall of OSQR, RWQR and OIQR for different number of walkers while fixing the TTL to 7. The experimental results are shown in Fig. 5.2 for deploying 1, 2 and 3 search walkers per query. One of the major goals of search is to achieve high recall rates with fewest search walkers. While our OSQR algorithm achieved 34.70%, 45.39%, and 53.49% recall rates on average for 1, 2 and 3 walkers, respectively,OIQR achieved only 7.22%, 12.31% and 16.26% recall and RWQR achieved only 4.25%, 8.13% and 12.12% for the respective number of walkers. Thus, OSQR shows an outstanding 380.29% improvement over OIQR and 717.15% improvement over RWQR even when walkers per query is as low as 1. The reason for better performance of OSQR comes from the fact that our semantic representation, PSI, is superior to their counterparts as it incorporates the twin notions of semantic richness (i.e., information content) of a peer and total qualified documents reachable through a peer per ontology concept. Therefore, in query routing, these semantic structures are better exploited by incorporating ontology knowledge to route queries to more promising peers. OIQR and RWQR, on the other hand, do not incorporate information content measures in query routing or query evaluation. Mere existence of query concepts in a given document is considered sufficient criterion for document relevance in OIQR.

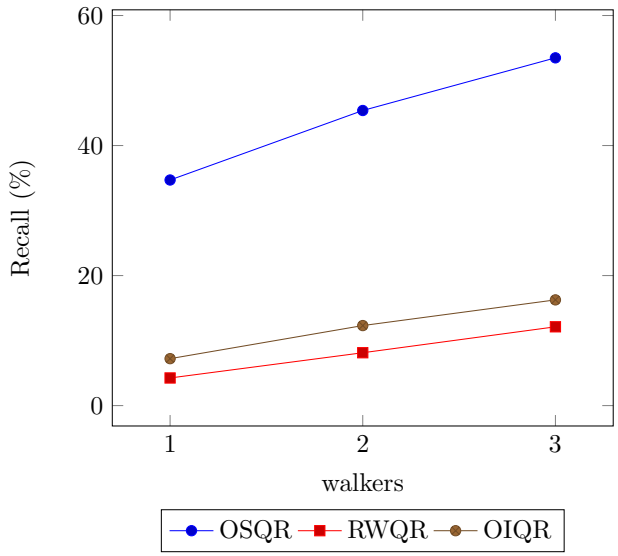


Figure (5.2) Average recall rate comparison between OSQR, RWQR and OIQR

5.6.2 Precision

Fig. 5.3 shows experimental results for precision for 1, 2, and 3 search walkers for OSQR, OIQR and RWQR algorithms. OSQR achieves a precision of 53.98%, 65.21% and 72.32% for 1, 2, and 3 search walkers, respectively. For the same number of walkers correspondingly, OIQR achieved only 10.58%, 16.00%, 19.59% precision and RWQR only, 6.54%, 10.96%, 14.70% precision. Thus, an average of 410.13% and 725.67% improvement in precision is observed in OSQR over OIQR and RWQR, respectively, for a single walker. The mechanisms of re-normalizing local document collections and updating PSIs contribute to this improvement in precision of OSQR over their counterparts. Re-normalizing the dataset essentially gears the normalized concept weight vectors of documents toward their globally normalized steady state values, reducing the possibility of incorrectly identifying documents irrelevant to a query as relevant in local query processing. Updating PSIs with information discovered from the network makes the semantic representations more accurate and precise over time leading to better judgment of peer selection in query routing. Fig.5.4 and Fig. 5.5 shows the recall and precision comparisons respectively for the three algorithms for different network sizes. The results demonstrate that OSQR clearly outperforms both OIQR and

RWQR in the P2P network at scale.

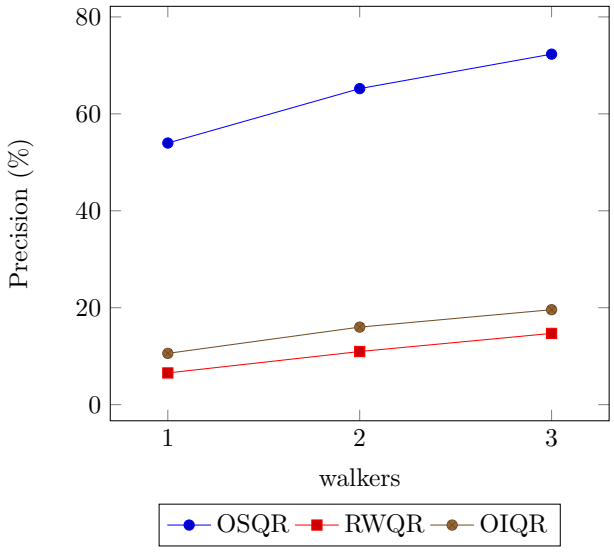


Figure (5.3) Average precision comparison between OSQR, RWQR and OIQR.

5.6.3 F-Measure

Fig. 5.6 depicts the F-Measure values which is considered a combined metric of retrieval performance for the three search algorithms for 1, 2, and 3 search walkers. F-Measure for OSQR is clearly superior to the other two comparison algorithms. OSQR achieves 0.42, 0.54, and 0.61 for F-Measure for 1, 2, and 3 walkers, respectively, whereas OIQR only achieves 0.09, 0.14 and 0.18 and RWQR 0.05, 0.09, and 0.13 for the respective number of walkers.

5.6.4 Search Cost

As mentioned before, our goal is to achieve better recall with fewer walkers deployed per query in order to minimize the cost of search. As experimental results show, OSQR achieves this goal successfully. The search cost was measured in terms of the average number of messages generated for a given query. Fig. 5.7 shows the experimental results for search cost. As shown in figure, the search cost of OSQR is twice that of OIQR and RWQR. This is because, in OSQR, a search response returns in the reverse query path back to the query originator compared to OIQR and RWQR, where search response message is directly

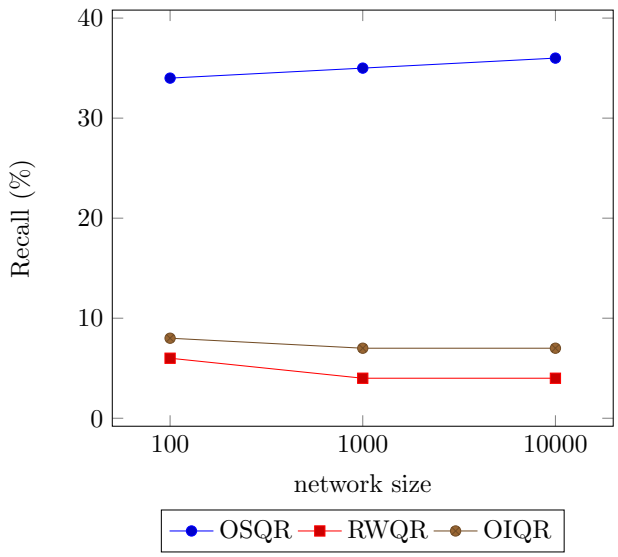


Figure (5.4) Average recall rate comparison between OSQR, OIQR and RWQR for varying network sizes

sent to query originator from the query response generator. Since each walker in OSQR travels TTL hops, the number of message exchanges per query is $2 \times k \times TTL$ for k-walker query. Note that this measure captures search traffic only. OIQR has a separate index updating mechanism which results in non-search based message traffic generation that is not represented in Fig. 5. Our OSQR algorithm, on the other hand, effectively utilizes search messages and their responses for PSI updating purposes without generating extra messages not intended for search purposes.

5.6.5 Hits per Query

Fig. 5.8 depicts the hits generated per search query for the three algorithms. It was observed that OSQR performs better compared to the other two algorithms. While OSQR generated 0.82 hits per single search walker, OIQR generated 0.15 hits and RWQR generated 0.12 hits only for one walker.

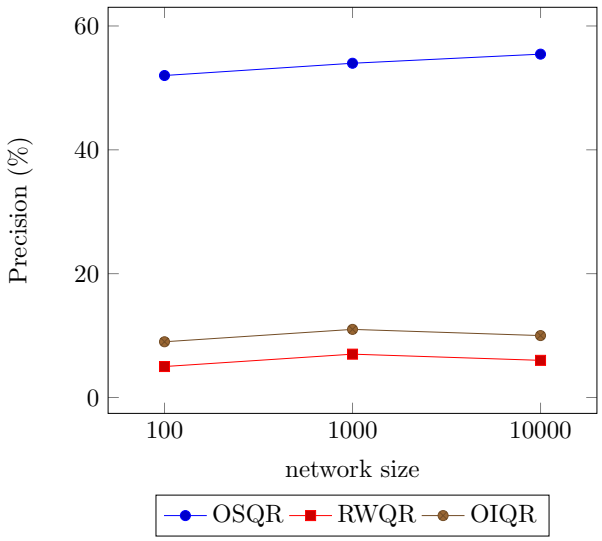


Figure (5.5) Average precision comparison between OSQR, OIQR and RWQR for varying network sizes

5.6.6 Time Complexity

Receipt of a query or a query response message results in a peer updating its MaxVector, which triggers re-normalizing its document collection. This in turn triggers recalculation of its PSIs for the affected concepts. Therefore, in the worst case, all $\mid C \mid$ concepts need to be updated. Therefore, for a given peer P, the time complexity of the algorithm whenever an renormalization is triggered is $O(\mid C \mid) \times O(\mid D(P) \mid) \times O(\mid PSI(P) \mid)$ where $\mid C \mid$ is the total number of concepts in the network, $\mid D(P) \mid$ the total number of documents in peer P and $\mid PSI(P) \mid$ the size of PSI of P or its local concept set. Fig. 5.5 shows how the updating cost of a PSI undergoes initial transitions and quickly diminishes over time. Simulation cycles here represent the time, while number of updates per cycle represents the PSI update cost. Each peer in the network issues 1 query per simulation cycle. The total number of PSI updates keeps increasing till 7th cycle. This is because peers start gaining knowledge from the network when queries start to be issued thus resulting in more updates per-cycle. In our simulation engine, a walker travels only 1 hop per cycle and, thus, 7 cycles are needed for a walker to travel TTL hops. Due to this intensity of new knowledge gained

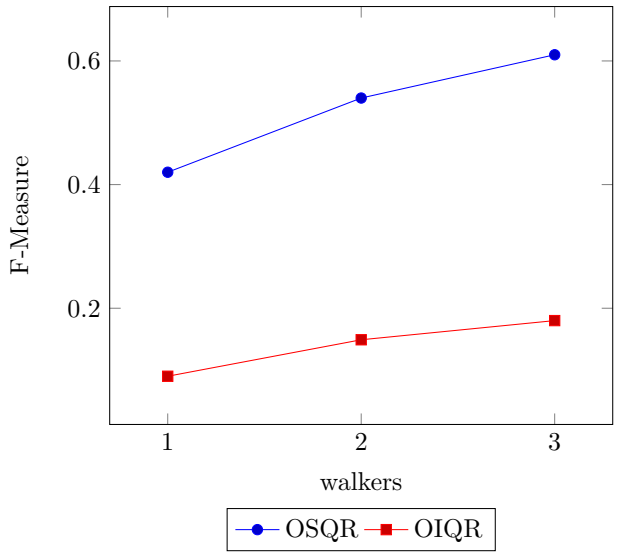


Figure (5.6) F-Measure comparison between OSQR, RWQR and OIQR.

from this forward updating, PSI update cost increases drastically. We noticed that, after 7th cycle, the total number of updates per cycle starts to diminish and fewer updates will be necessary when peers gain decreasingly fewer knowledge. At the end of the simulation, we observed that number of updates wear off to zero. Therefore, our algorithm effectively limits the computation time required to perform PSI updates while keeping the accuracy of PSI sufficiently high.

5.6.7 Space Complexity

Each peer P stores a MaxVector, PSI, neighbor PSIs, andthe local document concept weigh vectors CD(P). Given that the size of the local concept set is $\mid C(P) \mid$, MaxVector, PSI and each document concept weight vector in CD(P) takes O($\mid C(P) \mid$) space each, while each neighbor N's PSI at P takes O(C(N))space.

5.7 Summary

In this thesis, we propose a novel semantic query routing algorithm called OSQR for unstructured P2P networks. We effectively exploit semantic properties and their relationships

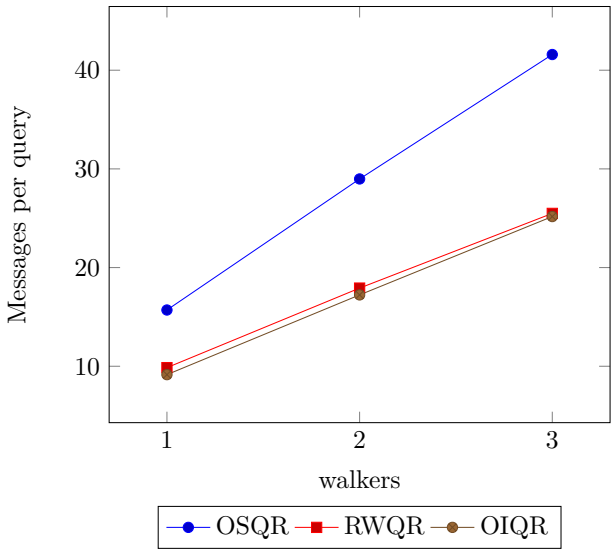


Figure (5.7) Search cost comparison between OSQR, OIQR and RWQR.

in a semantic hierarchy to achieve more informed query propagation as well as local search and best path for each concept to provide more relevant results. OSQR allows peers in the network to maintain limited soft state per neighbor. It is a simple and easy to implement algorithm. Experimental results show that OSQR outperforms Random Walk and Ontology Index based Query Routing, in terms of recall, precision and hits per query for comparable message costs. We plan to extend this work by developing a search algorithm that exploits a more complex ontology structure with relations other than hypernym/hyponym to perform query routing and query processing.

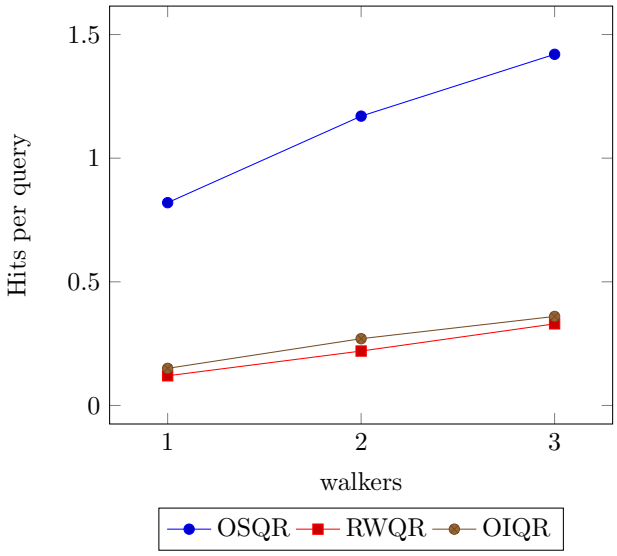


Figure (5.8) Hits Per Query comparison between OSQR, RWQR and OIQR.

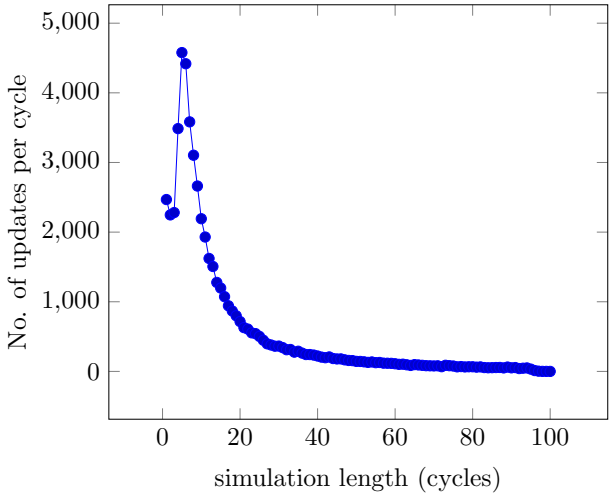


Figure (5.9) PSI updating cost of OSQR. The updating cost is calculated as the number of updates made per P2P simulation cycle

PART 6

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

6.1 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 [35], 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 [67], [29] are either computationally intensive [29] or index size varies largely with the dimensionality of the content [29]. 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 [**?**] 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 muti-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 chapter 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 [89], OLI [27], Ontsum [67] The related work most comparable to ours is OSQR [89], 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 muti-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 chapter is organized as follows: We present the related work in section **??** . The preliminary concepts used in the chapter are presented in section 6.2. The P2P system architecture including routing index construction and maintenance and query routing are given in section 6.3. In Section 6.4, we evaluate the proposed algorithms via simulation. Finally we conclude and summarize our work in section 6.5.

6.2 Preliminaries

6.2.1 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.

6.2.2 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.

6.2.3 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.

6.2.4 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 [86], [51] 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:

$$Similarity(D, Q) = \frac{D.Q}{\|D\| \, . \, \|Q\|} \tag{6.1}$$

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.

6.2.5 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 $(1 - e^{\frac{-kn}{m}})^k$.

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.

6.3 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

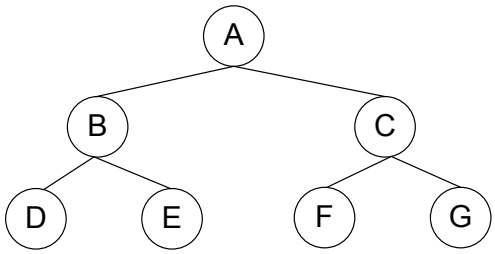


Figure (6.1) Ontology with seven concepts and IS-A relations between them.

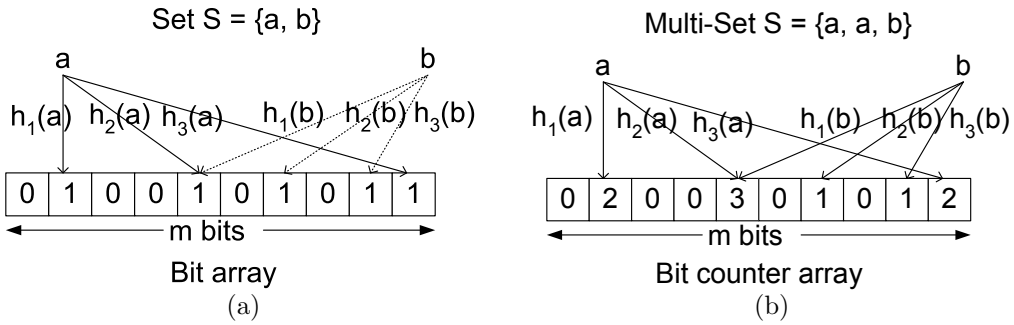


Figure (6.2) (a) A traditional Bloom filter with three hash functions. (b) A Spectral Bloom filter with three hash functions.

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.

6.3.1 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 $TSBF_P$?"* the $TSBF_P$ 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 [83, 94, 95] 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 6.3 shows a TSBF of a peer P for the ontology given in Figure 6.1. 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.

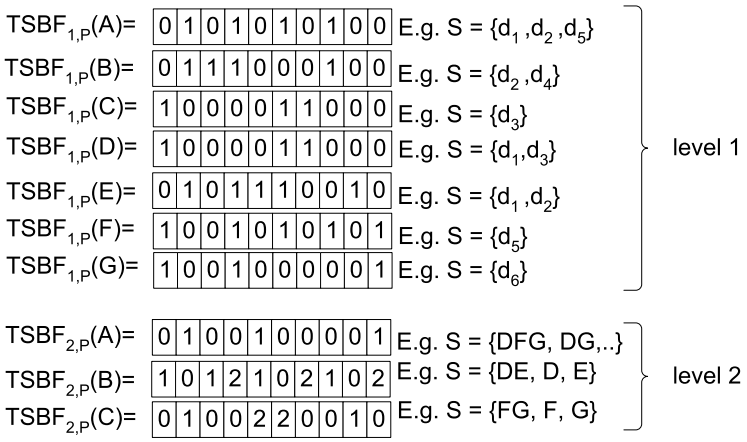


Figure (6.3) The TSBF for a peer P with documents $d_1d_2, d_3, d_4, d_5, d_6$. for ontology in Figure 6.1.

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.

TSBF Construction The TSBF of a peer P, $TSBF_P$, is constructed at start-up when peer joins the network for the fist time. Initially all the bits of level 1 bit arrays are initilized to false while the all the bits in level 2 are intialized 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 [**?**] 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.

6.3.2 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 $\langle key, value \rangle$ pairs where key is a neighbor N and value is a $TSBF_{P \to N}$, 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. $TSBF_{P \to N}$ generally represents an aggregation of a set of $TSBF_{P'}$ structures each belonging to a peer P' reachable through P.

Routing Index Creation and 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, $TSBF_{P \to N}$, at P's routing index is initially a replica of $TSBF_N$ 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 $TSBF_{P_i}$ 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 6.2 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.

$$TSBF_{x,expUnion}.(c)bit_i = F_{j=1}^{k}\left(\frac{TSBF_{x,N_j}(c).bit_i}{\alpha}\right) \tag{6.2}$$

The above design achieves our primary objective of efficiently maintaining probabilistic information about the content stored in the neighborhood. Algorithm 6.1 summarizes the routing index creation and updation procedure.

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 6.1 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 6.3. The level x is 1 or 2 and C_a is the ancestor concept of all descendant concepts c_j being aggregated. Similar to 6.2, F refers to bitwise or and summation of counts for level 1 and level 2 respectively.

$$TSBF_{x,union}.(c_a.bit_i = F_{j=1}^{k}(TSBF_{x,P\rightarrow N}.(c_j)bit_i) \tag{6.3}$$

When the need arises to compress a routing index, the priority first goes to combining

Algorithm 6.1 *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

1: $\triangledown$ *Create index entry for N for alive-ping message*
2: **if** $Msg.TYPE = ALIVE$ **then**
3: $TSBF_N \leftarrow TSBFList.get(0)$
4: **for each** $Concept\, c \in C$ **do**
5: **for** $i = 0$ to $TSBF_N(c).length - 1$ **do**
6: $index.TSBF_{1,P\rightarrow N}.(c).bit_i \leftarrow TSBF_{1,N}(c).bit_i$
7: $index.TSBF_{2,P\rightarrow N}.(c).bit_i \leftarrow TSBF_{2,N}(c).bit_i$
8: **end for**
9: **end for**
10: **end if**
11: $\triangledown$ *Remove index entry for N for leave-ping message*
12: **if** $Msg.TYPE = LEAVE$ **then**
13: $index.remove(N)$
14: **end if**
15: $\triangledown$ *Update index entry of N for information received through a search related message*
16: **if** $Msg.Type = QUERY\ or HIT\ or MISS$ **then**
17: $\triangledown$ *Determine concepts for which Routing index should be updated*
18: $Query\, q \leftarrow Msg.query$
19: $lca(q) \leftarrow least\, common\, ancestor\, concept\, of\, q$
20: $C_{update} \leftarrow$ 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: $\triangledown$ *Update Routing index entry for N*
24: **for each** $Concept\, c \in C_{update}$ **do**
25: $\triangledown$ *Calculate a new TSBF entry for N based on TSBFList*
26: $TSBF_{1,N'}(c) \leftarrow$ calculate using Equation 6.2
27: $TSBF_{2,N'}(c) \leftarrow$ calculate using Equation 6.2
28: **if** $TSBF_{1,N'}(c).bit_i > index.TSBF_{1,P\rightarrow N}(c).bit_i$ **then**
29: $index.TSBF_{1,P\rightarrow N}(c).bit_i \leftarrow TSBF_{1,N'}(C).bit_i$
30: **end if**
31: **if** $TSBF_{2,N'}(c).bit_i > index.TSBF_{2,P\rightarrow N}(c).bit_i$ **then**
32: $index.TSBF_{2,P\rightarrow N}(c).bit_i \leftarrow TSBF_{2,path}(C).bit_i$
33: **end if**
34: **end for**
35: **end if**

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.

6.3.3 Query Routing

he query routing algorithm is summarized in Algorithm 6.2. 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.

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 cardinatily(i.e. number of elements) of an original set solely based on its Bloom filter bit array [83,94,95]. For our work we used the work presented by authors of [83].

Given a set S and its Bloom Filter BF_S with m bits out of which t are true bits, and k hash functions [83] defines the cardinality of a a set as follows:

$$|S| = \frac{ln\left(1 - \frac{t}{m}\right)}{k \times ln\left(1 - \frac{1}{m}\right)} \tag{6.4}$$

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 $S_{\bigcup} = S_1 \bigcup S_2 \bigcup ... \bigcup S_n$ (needed for OR query routing) authors simply propose to take the bitwise OR of the representative Bloom filter bit arrays and apply equation 6.4 to the resulting Bloom filter.

Estimating cardinatily 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 6.5 to calculate the intersection of sets $S_{\bigcap} = S_1 \bigcap S_2 \bigcap ... \bigcap S_n$:

$$\left|\bigcap_{i=1}^{n} S_i\right| = \sum_{k=1}^{n} (-1)^{k+1} \left(\sum_{1 \leq i < k \leq n}^{v} \left| S_i \bigcup ... \bigcup S_k \right| \right) \tag{6.5}$$

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 6.5 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 6.5 thus computing the cardinality of intersection for a neighbor for a given query.

6.4 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.

6.4.1 Simulation Methodology

The parameters for the the simulation TSBF used in BSI are listed in table 6.1. The simulations use a default 1024 peer unstructured network, unless noted otherwise[1]. 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 [89] and Random Walk [79]. 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

[1]The routing tables in BSI make the simulations very memory intensive and do not scale to large sizes easily. For the sake of uniformity, we simulate a network of 1024 nodes for all search protocols.

Algorithm 6.2 *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: $\triangledown$ ***Found target Object***
5: **if** $Q.TTL \leq 0$ **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: $\triangledown$ ***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: $\triangledown$ ***Estimate score from*** $index.TSBF_{1,P \rightarrow N}$
18: **else**
19: $QueryBFList_N \leftarrow \phi$
20: **for each** Concept c in Q **do**
21: $QueryBFList_N.add(index.TSBF_{1,P \rightarrow N}.(c))$
22: **end for**
23: $Score_N \leftarrow$ calculate using Equation 6.5 with $QueryBFList_N$ as input
24: **end if**
25: Sort $Candidates(Q)$ based on scores
26: $Selected(Q) \leftarrow$ Neighbor in $Candidates(Q)$ with highest score
27: **end for**
28: Return $Selected(Q)$ and R
29: **end if**

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 random neighbor. The neighbor selection mechanism does not require a peer to maintain any local knowledge about its neighborhood. 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.

6.4.2 Results

The primary performance results can 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 6.4.2. Simulations over different query lengths and different filter sizes demonstrate the scalability of BSI for multi-term query routing in sections 6.4.2 and 6.4.2 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

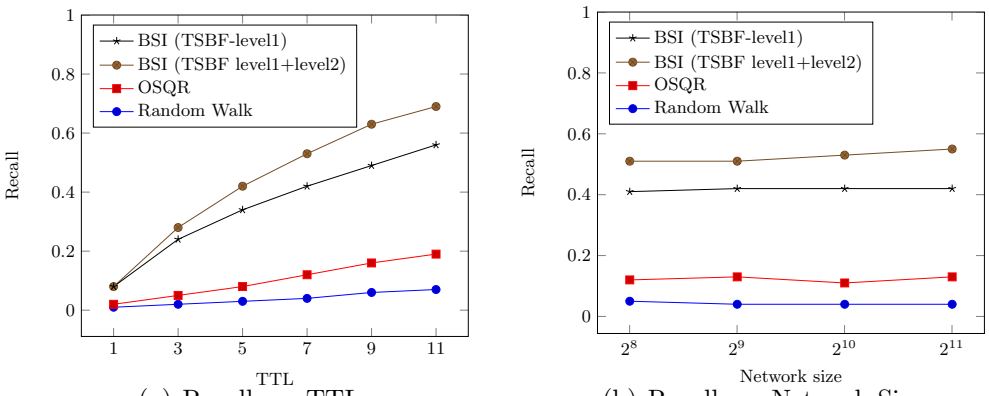
search efficiency in section 6.4.2 demonstrates that the performance improvement in recall is achieved at an attractive two-fold improvement in search traffic over OSQR.

Table (6.1) 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

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. Fig. 6.4(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 performace improvement of BSI(TSBF-level1+level2) over Random walk was observed to be an ourstanding 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 Fig. 6.4(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 (6.4) Recall

Effect of Query Length One of the key design objectives of BSI is to achieve high recalls for queries with more than single concept. Fig. 6.5 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.

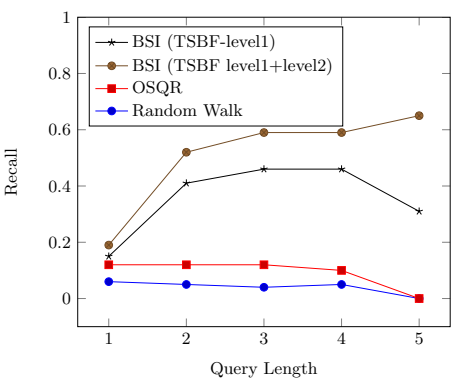
Random walk does not use any intelligence in neighbor selection mechanism. Peers randomly select one if their neighbors to forwrd queries. BSI and OSQR estimate the number of documents reachable thorough a peer for a query as its relevance score in neighbor selection process. An OSQR peer only maintains 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.

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 thorugh a neighbor, thereby providing a more compact and high quality information summarization leading to more accurate estimations of number of documents reachable for queries.

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.

Table (6.2) Storage Overhead

Algorithm	Filter Size(bits)		
	100	300	500
OSQR	10752.01	10752.01	10752.01
BSI (TSBF level1+level2)	4875.00	14625.00	24375.00
BSI (TSBF-level1)	4800.00	14400.00	24000.00
BSI (TSBF-level2)	56.25	168.75	281.25



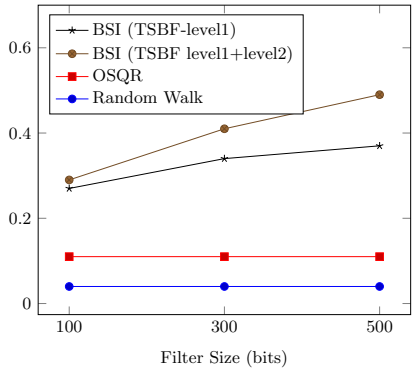


Figure (6.5) Recall vs. Query Length Figure (6.6) Recall vs. Filter Size

Using TSBFs for local knowledge maintenance results in substantial increase in recall at low storage cost. As shown in Fig. 6.6 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. 6.2 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 **??** or to eliminate one of the levels of TSBFs stored in routing indices to observe their storage restrictions.

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 level1+level2)and BSI (TSBF-level1) respectively. This corresponded to a 70.35%, and 95.46% improvement over OSQR in traffic cost for BSI (TSBF level1+level2)and BSI (TSBF-level1)respectively. Random Walk search message consumed only 1.02KB banwidth 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 heavy weight routing indices of visited peers of a query as well as the global statistics of maximum information content encountered so far for the large document base distributed in the P2P network per ontology concept by each visited peer. Queries in BSI on the other hand transfers much more space efficient TSBFs only relevant to the queried ontology concepts thus significantly reducing the bandwidth consumption.

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 between recall and search cost. In Fig. 6.8 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 OSQR and Random Walk, and BSI is capable of producing high recall at much lower message cost.

In conclusion from the experimental results we can draw the conclusion that with respect to recall rate and search cost, that our search protocol is superior than the OSQR.

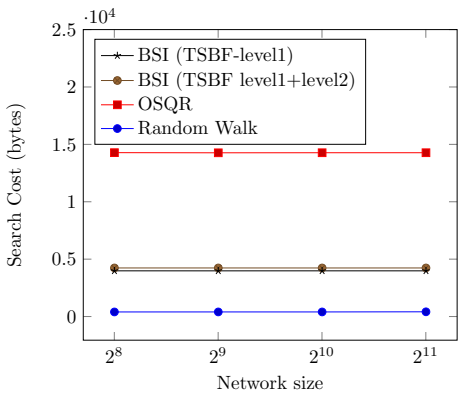


Figure (6.7) Search Cost

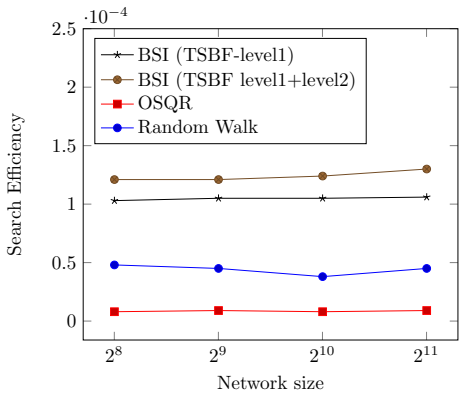


Figure (6.8) Search Efficiency

6.5 Conclusion and Future Work

Routing in unstructured P2P networks is a challenging problem. Unfortunately the prerequisites for efficient multi-term query routing such as large size multi-keyword based indexing is too costly in peer-to-peer networks. In this work, we have explored the approach of compact representation of large concept-combination based indices through the use of Bloom filters to achieve minimal loss of information. Our Bloom filter based routing index quantifies the strength of a peer's neighborhood for individual queries and then use this information for forwarding queries. The simple yet powerful design of BSI makes it easy implement and allows easy adapting to dynamic P2P conditions. Simulation based comparison with state-of-the art query routing mechanisms, under a wide variety of scenarios establish the performance advantages of BSI. While providing a set of mechanisms for query routing, BSI imposes little restriction on the possible policies that can be implemented. The BSI search framework presents interesting possibilities of implementing high level semantics of trust, reliability, etc. using routing and forwarding policies. These are interesting issues that merit further exploration.

PART 7

CONCLUSIONS AND FUTURE DIRECTIONS

In this chapter, we summarize the research presented in this dissertation, discuss our major contributions and their significance, and point out directions for future work.

7.1 Summary

In the last few years, the research community has provided a plethora of powerful tools in the area of distributed communications. The interest in P2P computing produced a variety of systems and schemes that facilitate the two important primitives in large decentralized environments: Content sharing and open communication.

The work discussed in this dissertation provides an integrated framework for full-text federated search of large and distributed document collections using unstructured P2P networks as the search layer. Our dissertation focuses exclusively on providing adaptive, bandwidth-efficient protocols for data search, retrieval and knowledge management in unstructured overlays with a special focus on combining the power of semantic computing and P2P networks to provide semantic search capabilities in P2P networks. Comprehensive evaluations measure the performance of the suit of query routing replication and topology optimization algorithms and compare it with existing common alternatives. Experimental results provide strong empirical evidence for the effectiveness of the approaches proposed in this dissertation for full-text federated search in P2P networks.

7.2 Contributions

This dissertation made three main contributions, at the architectural, algorithmic and implementation level. We proposed a suit of indexing architectures (SPUN, OSQR)that clearly allows peers to efficiently summarize the knowledge about their neighborhoods is

a simple yet power manner. While the SPUN indexing architecture [96] allows a peer to predict the probability of success of a neighbor based on their past interactions, the indexing scheme proposed in OSQR [89] allows to condense multi-dimensional data to a constant and small size single dimension using an automatic semantic annotation of resources. The OSI index structure on the other hand present a space-efficient concept-set based indexing mechanism that allow peers to represent large volume of concept-set combinations along with their relevance strengths in a compact manner, thereby improving the search quality of multi-concept queries. Our indexing schemes allows us to reuse existing algorithms, to tailor the index structure to different application requirements, and to evaluate new and existing algorithms in the same code base. We introduced R-SPUN, a new replication algorithm designed to improve efficiency of rare object search. We also introduce SAS, a novel semantic clustering algorithm that reorganizes the original network topology to mimic a reference ontology structure, thereby allowing peers to exploit ontology relationships to perform inter-cluster and intra-cluster routing. SAS provides provable guarantees on search performance in a stable system. We performed extensive experiments on Peersim simulator by simulating the unstructured overay environment based on Gnutella performance measurement studies and using Reuters news-wire data set. Our experimental evaluation shows that our indexing, replication and clustering algorithms outperform existing state-of-the-art counterparts, and that it maintains its excellent search performance with low maintenance costs in a dynamic P2P system.

7.3 Future Work

The work presented in this dissertation represents just the first steps towards building a fully functional P2P database systems. We plan to address the issues of data management and query routing in presence of semantic heterogeneity where peers describe their resources using local ontologies. We will also look into the possibility of merging the powers of cloud computing and P2P. This mixed model could be applied to several different applications, including backup, storage, content distribution and search allowing exploiting (free) user

resources whenever possible, reducing the bill to be payed to cloud provider.

REFERENCES

[1] "Gnutella website," http://www.gnutella.com/, Jan. 2007. [Online]. Available: http://www.gnutella.com/

[2] (2006, Jun.) Kazaa.com Website. http://www.kazaa.com/. [Online]. Available: http://www.kazaa.com/us/index.htm

[3] "What Is P2P ... And What Isnt," http://www.OpenP2P.com/, 2000. [Online]. Available: http://www.OpenP2P.com/

[4] S. Inc, "The impact of file sharing on service provider networks," Tech. Rep.

[5] H. Rostami, J. Habibi, and E. Livani, "Semantic routing of search queries in p2p networks," *J. Parallel Distrib. Comput.*, vol. 68, no. 12, pp. 1590–1602, Dec. 2008. [Online]. Available: http://dx.doi.org/10.1016/j.jpdc.2008.06.005

[6] K. Aberer, P. Cudré-Mauroux, and M. Hauswirth, "The chatty web: emergent semantics through gossiping," in *Proceedings of the 12th international conference on World Wide Web*, ser. WWW '03. New York, NY, USA: ACM, 2003, pp. 197–206. [Online]. Available: http://doi.acm.org/10.1145/775152.775180

[7] K. Aberer, P. Cudre-Mauroux, M. Hauswirth, and T. V. Pelt, "Gridvine: Building internet-scale semantic overlay networks," in *In International Semantic Web Conference*, 2004, pp. 107–121.

[8] A. U. Ahmed, T. Shahid, H. Chen, and Q. Jin, "A hybrid p2p search engine for social learning," in *Proceedings of the 2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing*, ser. ITHINGSCPSCOM '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 564–569. [Online]. Available: http://dx.doi.org/10.1109/iThings/CPSCom.2011.111

[9] M. Arumugam, A. Sheth, and I. B. Arpinar, "Towards peer-to-peer semantic web: A distributed environment for sharing semantic knowledge on the web," Tech. Rep., 2001.

[10] B. T. Loo, J. M. Hellerstein, R. Huebsch, S. Shenker, and I. Stoica, "Enhancing p2p file-sharing with an internet-scale query processor," in *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30*, ser. VLDB '04. VLDB Endowment, 2004, pp. 432–443. [Online]. Available: http://dl.acm.org/citation.cfm?id=1316689.1316728

[11] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM '01. New York, NY, USA: ACM, 2001, pp. 149–160. [Online]. Available: http://doi.acm.org/10.1145/383059.383071

[12] A. I. T. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, ser. Middleware '01. London, UK, UK: Springer-Verlag, 2001, pp. 329–350. [Online]. Available: http://dl.acm.org/citation.cfm?id=646591.697650

[13] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," *SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 4, pp. 161–172, Aug. 2001. [Online]. Available: http://doi.acm.org/10.1145/964723.383072

[14] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and replication in unstructured peer-to-peer networks," in *Proceedings of the 16th international conference on Supercomputing*, ser. ICS '02. New York, NY, USA: ACM, 2002, pp. 84–95. [Online]. Available: http://doi.acm.org/10.1145/514191.514206

[15] V. Kalogeraki, D. 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*, ser. CIKM '02. New York, NY, USA: ACM, 2002, pp. 300–307. [Online]. Available: http://doi.acm.org/10.1145/584792.584842

[16] B. Yang and H. Garcia-Molina, "Improving search in peer-to-peer networks," in *Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)*, ser. ICDCS '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 5–. [Online]. Available: http://dl.acm.org/citation.cfm?id=850928.851859

[17] A. Crespo and H. Garcia-Molina, "Routing indices for peer-to-peer systems," in *Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)*, ser. ICDCS '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 23–. [Online]. Available: http://dl.acm.org/citation.cfm?id=850928.851858

[18] D. Tsoumakos and N. Roussopoulos, "A comparison of peerto-peer search methods," in *In WebDB*, 2003.

[19] B. Yang and H. Garcia-molina, "Efficient search in peer-to-peer networks," 2002.

[20] A. Crespo and H. Garcia-Molina, "Semantic overlay networks for p2p systems," in *Proceedings of the Third international conference on Agents and Peer-to-Peer Computing*, ser. AP2PC'04. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 1–13. [Online]. Available: http://dx.doi.org/10.1007/11574781_1

[21] K. Sripanidkulchai, B. Maggs, and H. Zhang, "Efficient content location using interest-based locality in peer-to-peer systems," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 3, march-3 april 2003, pp. 2166 – 2176 vol.3.

[22] M. Bawa, G. S. Manku, and P. Raghavan, "Sets: search enhanced by topic segmentation," in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, ser. SIGIR

'03. New York, NY, USA: ACM, 2003, pp. 306–313. [Online]. Available: http://doi.acm.org/10.1145/860435.860491

[23] Y. Zhu and Y. Hu, "Enhancing search performance on gnutella-like p2p systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 17, no. 12, pp. 1482–1495, Dec. 2006. [Online]. Available: http://dx.doi.org/10.1109/TPDS.2006.173

[24] J. Huang, X. Li, and J. Wu, "A class-based search system in unstructured p2p networks," in *Advanced Information Networking and Applications, 2007. AINA '07. 21st International Conference on*, 2007, pp. 76–83.

[25] C. Tang, Z. Xu, and M. Mahalingam, "psearch: information retrieval in structured overlays," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 89–94, Jan. 2003. [Online]. Available: http://doi.acm.org/10.1145/774763.774777

[26] M. Li, W.-C. Lee, and A. Sivasubramaniam, "Semantic small world: An overlay network for peer-to-peer search," in *Proceedings of the 12th IEEE International Conference on Network Protocols*, ser. ICNP '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 228–238. [Online]. Available: http://dl.acm.org/citation.cfm?id=1025124.1025895

[27] H. Rostami, J. Habibi, H. Abolhassani, M. Amirkhani, and A. Rahnama, "An ontology based local index in p2p networks," in *Proceedings of the Second International Conference on Semantics, Knowledge, and Grid*, ser. SKG '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 11–. [Online]. Available: http://dx.doi.org/10.1109/SKG.2006.25

[28] M. Schlosser, M. Sintek, S. Decker, and W. Nejdl, "Hypercup - shaping up peer-to-peer networks," Tech. Rep., 2002.

[29] 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*, 2008, pp. 71–76.

[30] M. Cai and M. Frank, "Rdfpeers: a scalable distributed rdf repository based on a structured peer-to-peer network," in *Proceedings of the 13th international conference on World Wide Web*, ser. WWW '04. New York, NY, USA: ACM, 2004, pp. 650–657. [Online]. Available: http://doi.acm.org/10.1145/988672.988760

[31] A. Bonifati, U. Matrangolo, A. Cuzzocrea, and M. Jain, "Xpath lookup queries in p2p networks," in *Proceedings of the 6th annual ACM international workshop on Web information and data management*, ser. WIDM '04. New York, NY, USA: ACM, 2004, pp. 48–55. [Online]. Available: http://doi.acm.org/10.1145/1031453.1031464

[32] W. S. Ng, B.-C. Ooi, K.-L. Tan, and A. Zhou, "Peerdb: a p2p-based system for distributed data sharing," in *Data Engineering, 2003. Proceedings. 19th International Conference on*, 2003, pp. 633–644.

[33] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," *Commun. ACM*, vol. 18, no. 11, pp. 613–620, Nov. 1975. [Online]. Available: http://doi.acm.org/10.1145/361219.361220

[34] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970. [Online]. Available: http://doi.acm.org/10.1145/362686.362692

[35] (2006) Napster website. http://www.napster.com/. [Online]. Available: http://www.napster.com/

[36] "iMesh website," http://www.imesh.com/, 1999. [Online]. Available: http://www.imesh.com/

[37] F. M. Cuenca-acuna, C. Peery, R. P. Martin, and T. D. Nguyen, "Planetp: Using gossiping to build content addressable peer-to-peer information sharing communities." IEEE Press, 2003.

[38] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer, "Improving collection selection with overlap awareness in p2p search engines," in *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '05. New York, NY, USA: ACM, 2005, pp. 67–74. [Online]. Available: http://doi.acm.org/10.1145/1076034.1076049

[39] T. Suel, C. Mathur, J. wen Wu, J. Zhang, A. Delis, M. Kharrazi, X. Long, and K. Shanmugasundaram, "Odissea: A peer-to-peer architecture for scalable web search and information retrieval," in *In WebDB*, 2003.

[40] Y. Mei, J. Wang, Q. Wang, X. Liu, Z. Zhao, and Y. Zhang, "Semantic-based query routing in p2p social networks," in *Wireless Communications, Networking and Information Security (WCNIS), 2010 IEEE International Conference on*, 2010, pp. 674–678.

[41] W. Shen, S. Su, K. Shuang, F. Yang, and J. Xia, "An efficient search mechanism in unstructured p2p networks based on semantic group," in *Proceedings of the 2010 10th IEEE International Conference on Computer and Information Technology*, ser. CIT '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 2982–2986. [Online]. Available: http://dx.doi.org/10.1109/CIT.2010.497

[42] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: a scalable wide-area web cache sharing protocol," *IEEE/ACM Trans. Netw.*, vol. 8, no. 3, pp. 281–293, Jun. 2000. [Online]. Available: http://dx.doi.org/10.1109/90.851975

[43] K. Bratbergsengen, "Hashing methods and relational algebra operations," in *Proceedings of the 10th International Conference on Very Large Data Bases*, ser. VLDB '84. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1984, pp. 323–333. [Online]. Available: http://dl.acm.org/citation.cfm?id=645912.671296

[44] J. K. Mullin, "Optimal semijoins for distributed database systems," *IEEE Trans. Softw. Eng.*, vol. 16, no. 5, pp. 558–560, May 1990. [Online]. Available: http://dx.doi.org/10.1109/32.52778

[45] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," in *Internet Mathematics*, 2002, pp. 636–646.

[46] J. W. Byers, J. Considine, M. Mitzenmacher, and S. Rost, "Informed content delivery across adaptive overlay networks," *IEEE/ACM Trans. Netw.*, vol. 12, no. 5, pp. 767–780, Oct. 2004. [Online]. Available: http://dx.doi.org/10.1109/TNET.2004.836103

[47]

[48] A. Kumar, J. Xu, and E. W. Zegura, "Efficient and scalable query routing for unstructured peer-to-peer networks," in *INFOCOM*, 2005, pp. 1162–1173.

[49] J. Ledlie, J. M. Taylor, L. Serban, and M. Seltzer, "Self-organization in peer-to-peer systems," in *Proceedings of the 10th workshop on ACM SIGOPS European workshop*, ser. EW 10. New York, NY, USA: ACM, 2002, pp. 125–132. [Online]. Available: http://doi.acm.org/10.1145/1133373.1133397

[50] J. Li, B. Loo, J. Hellerstein, M. Kaashoek, D. Karger, and R. Morris, "On the feasibility of peer-to-peer web indexing and search," in *Peer-to-Peer Systems II*, ser. Lecture Notes in Computer Science, M. Kaashoek and I. Stoica, Eds. Springer Berlin Heidelberg, 2003, vol. 2735, pp. 207–215. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-45172-3_19

[51] M. W. Berry, Z. Drmac, and E. R. Jessup, "Matrices, vector spaces, and information retrieval," *SIAM Rev.*, vol. 41, no. 2, pp. 335–362, Jun. 1999. [Online]. Available: http://dx.doi.org/10.1137/S0036144598347035

[52] C. Xu and Y. Zhang, "Ontology based p2p semantic search routing algorithm," in *Networking, Sensing and Control (ICNSC), 2010 International Conference on*, 2010, pp. 689–692.

[53] J. Cai, X. Shao, and W. Ma, "Ontology driven semantic search over structure p2p

network," in *Hybrid Intelligent Systems, 2009. HIS '09. Ninth International Conference on*, vol. 3, 2009, pp. 29–34.

[54] W. Ma, G. Wang, and X. Liu, "Scalable semantic search with hybrid concept index over structure peer-to-peer network," in *Grid and Cooperative Computing, 2007. GCC 2007. Sixth International Conference on*, 2007, pp. 42–48.

[55] P. C. G. da Costa, K. B. Laskey, and T. Lukasiewicz, *Uncertainty Representation and Reasoning in the Semantic Web*. Information Science Reference, October 2008, pp. 315–340. [Online]. Available: http://dx.doi.org/10.4018/978-1-60566-112-4.ch013

[56] L. Zhang, "Processing query in p2p xml data sources," in *Intelligence Science and Information Engineering (ISIE), 2011 International Conference on*, 2011, pp. 526–529.

[57] N. Koudas, M. Rabinovich, D. Srivastava, and T. Yu, "Routing xml queries," in *Data Engineering, 2004. Proceedings. 20th International Conference on*, 2004, pp. 844–.

[58] L. Ali and G. Lausan, "Evaluating sparql subqueries over p2p overlay networks," in *Database and Expert Systems Applications (DEXA), 2012 23rd International Workshop on*, 2012, pp. 272–276.

[59] B. F. Cooper, "An optimal overlay topology for routing peer-to-peer searches," in *Proceedings of the ACM/IFIP/USENIX 6th international conference on Middleware*, ser. Middleware'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 82–101. [Online]. Available: http://dx.doi.org/10.1007/11587552_5

[60] Y. Wang, W. Li, J. Chen, S. Lu, F. Wang, and D. Chen, "Rsb-topo: A topology adaptation algorithm for unstructured p2p networks," in *Parallel Processing Workshops, 2007. ICPPW 2007. International Conference on*, 2007, pp. 73–73.

[61] Q. Lv, S. Ratnasamy, and S. Shenker, "Can heterogeneity make gnutella scalable?" in *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, ser.

IPTPS '01. London, UK, UK: Springer-Verlag, 2002, pp. 94–103. [Online]. Available: http://dl.acm.org/citation.cfm?id=646334.687805

[62] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making gnutella-like p2p systems scalable," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM '03. New York, NY, USA: ACM, 2003, pp. 407–418. [Online]. Available: http://doi.acm.org/10.1145/863955.864000

[63] J. Hu, M. Li, W. Zheng, D. Wang, N. Ning, and H. Dong, "Smartboa: Constructing p2p overlay network in the heterogeneous internet using irregular routing tables," in *Peer-to-Peer Systems III*, ser. Lecture Notes in Computer Science, G. Voelker and S. Shenker, Eds. Springer Berlin Heidelberg, 2005, vol. 3279, pp. 278–287. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-30183-7_27

[64] M. Waldvogel and R. Rinaldi, "Efficient topology-aware overlay network," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 101–106, Jan. 2003. [Online]. Available: http://doi.acm.org/10.1145/774763.774779

[65] C. Wang and L. Xiao, "An effective p2p search scheme to exploit file sharing heterogeneity," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 2, pp. 145–157, Feb. 2007. [Online]. Available: http://dx.doi.org/10.1109/TPDS.2007.20

[66] V. Carchiolo, M. Malgeri, G. Mangioni, and V. Nicosia, "Emerging structures of p2p networks induced by social relationships," *Comput. Commun.*, vol. 31, no. 3, pp. 620–628, Feb. 2008. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2007.08.016

[67] J. Li and S. 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*, ser. AINA '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 94–101. [Online]. Available: http://dx.doi.org/10.1109/AINA.2007.104

[68] G. Chen, C. P. Low, and Z. Yang, "Enhancing search performance in unstructured p2p networks based on users' common interest," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 6, pp. 821–836, Jun. 2008. [Online]. Available: http://dx.doi.org/10.1109/TPDS.2008.42

[69] H. Jin, X. Ning, and H. Chen, "Efficient search for peer-to-peer information retrieval using semantic small world," in *Proceedings of the 15th international conference on World Wide Web*, ser. WWW '06. New York, NY, USA: ACM, 2006, pp. 1003–1004. [Online]. Available: http://doi.acm.org/10.1145/1135777.1135986

[70] K. Y. Hui, J. C. Lui, and D. K. Yau, "Small-world overlay {P2P} networks: Construction, management and handling of dynamic flash crowds," *Computer Networks*, vol. 50, no. 15, pp. 2727 – 2746, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128605003609

[71] J. Kleinberg, "The small-world phenomenon: an algorithmic perspective," in *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, ser. STOC '00. New York, NY, USA: ACM, 2000, pp. 163–170. [Online]. Available: http://doi.acm.org/10.1145/335305.335325

[72] H. Zhang, A. Goel, and R. Govindan, "Using the small-world model to improve freenet performance," *Computer Networks*, vol. 46, no. 4, pp. 555 – 574, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128604001550

[73] E. Leontiadis, V. Dimakopoulos, and E. Pitoura, "Creating and maintaining replicas in unstructured peer-to-peer systems," in *Euro-Par 2006 Parallel Processing*, ser. Lecture Notes in Computer Science, W. Nagel, W. Walter, and W. Lehner, Eds. Springer Berlin Heidelberg, 2006, vol. 4128, pp. 1015–1025. [Online]. Available: http://dx.doi.org/10.1007/11823285_107

[74] M. A. Nascimento, "Peer-to-peer: harnessing the power of disruptive technologies,"

SIGMOD Rec., vol. 32, no. 2, pp. 57–58, Jun. 2003. [Online]. Available: http://doi.acm.org/10.1145/776985.776996

[75] H. Yamamoto, D. Maruta, and Y. Oie, "Replication methods for load balancing on distributed storages in p2p networks," in *Proceedings of the The 2005 Symposium on Applications and the Internet*, ser. SAINT '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 264–271. [Online]. Available: http://dx.doi.org/10.1109/SAINT.2005.53

[76] E. Leontiadis, V. V. Dimakopoulos, and E. Pitoura, "E.: Creating and maintaining replicas in unstructured peer-to-peer systems," In 12th International Euro-Par Conference on Parallel Processing, Tech. Rep., 2006.

[77] D. Tsoumakos and N. Roussopoulos, "An adaptive probabilistic replication method for unstructured p2p networks," in *Proceedings of the 2006 Confederated international conference on On the Move to Meaningful Internet Systems: CoopIS, DOA, GADA, and ODBASE - Volume Part I*, ser. ODBASE'06/OTM'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 480–497. [Online]. Available: http://dx.doi.org/10.1007/11914853_29

[78] J. Kangasharju, K. W. Ross, S. Antipolls, and D. A. Turner, "Optimal content replication in p2p communities," Tech. Rep., 2002.

[79] C. Gkantsidis, M. Mihail, and A. Saberi, "Random walks in peer-to-peer networks: algorithms and evaluation," *Perform. Eval.*, vol. 63, no. 3, pp. 241–263, Mar. 2006. [Online]. Available: http://dx.doi.org/10.1016/j.peva.2005.01.002

[80] D. Tsoumakos and N. Roussopoulos, "Adaptive probabilistic search for peer-to-peer networks," in *Peer-to-Peer Computing, 2003. (P2P 2003). Proceedings. Third International Conference on*, 2003, pp. 102–109.

[81] L. A. Adamic and B. A. Huberman, "Zipf's law and the internet," *Glottometrics*, vol. 3, pp. 143–150, 2002.

[82] J. Wu and H. Li, "On calculating connected dominating set for efficient routing in ad hoc wireless networks," in *Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications*, ser. DIALM '99. New York, NY, USA: ACM, 1999, pp. 7–14. [Online]. Available: http://doi.acm.org/10.1145/313239.313261

[83] O. Papapetrou, W. Siberski, and W. Nejdl, "Cardinality estimation and dynamic length adaptation for bloom filters," *Distrib. Parallel Databases*, vol. 28, no. 2-3, pp. 119–156, Dec. 2010. [Online]. Available: http://dx.doi.org/10.1007/s10619-010-7067-2

[84] S. Cohen and Y. Matias, "Spectral bloom filters," in *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '03. New York, NY, USA: ACM, 2003, pp. 241–252. [Online]. Available: http://doi.acm.org/10.1145/872757.872787

[85] M. Harish, N. Anandavelu, N. Anbalagan, G. S. Mahalakshmi, and T. V. Geetha, "Design and analysis of a game theoretic model for p2p trust management," in *Proceedings of the 4th International Conference on Distributed Computing and Internet Technology (ICDCIT'07)*, ser. Lecture Notes in Computer Science, T. Janowski and H. Mohanty, Eds., vol. 4882. Springer, Dec. 2007, pp. 110–115.

[86] M. Bendersky and W. B. Croft, "Discovering key concepts in verbose queries," in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '08. New York, NY, USA: ACM, 2008, pp. 491–498. [Online]. Available: http://doi.acm.org/10.1145/1390334.1390419

[87] Y. Li, Z. Bandar, and D. McLean, "An approach for measuring semantic similarity between words using multiple information sources," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 15, no. 4, pp. 871 – 882, july-aug. 2003.

[88] H. Schütze and C. Silverstein, "Projections for efficient document clustering," in *Proceedings of the 20th annual international ACM SIGIR conference on Research and*

development in information retrieval, ser. SIGIR '97. New York, NY, USA: ACM, 1997, pp. 74–81. [Online]. Available: http://doi.acm.org/10.1145/258525.258539

[89] R. Dissanayaka, S. Navathe, and S. 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*, ser. HiPC '12. IEEE, 2012.

[90] D. D. Lewis, "Reuters dataset," http://trec.nist.gov/data/reuters/reuters.htm, 2004.

[91] A. Montresor and M. Jelasity, "PeerSim: A scalable P2P simulator," in *Proc. of the 9th Int. Conference on Peer-to-Peer (P2P'09)*, Seattle, WA, Sep. 2009, pp. 99–100.

[92] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "A measurement study of peer-to-peer file sharing systems," 2002.

[93] H. Jin and Y. Yu, "Semrex: a semantic peer-to-peer scientific references sharing system," in *Telecommunications, 2006. AICT-ICIW '06. International Conference on Internet and Web Applications and Services/Advanced International Conference on*, 2006, pp. 97–97.

[94] L. Michael, W. Nejdl, O. Papapetrou, and W. Siberski, "Improving distributed join efficiency with extended bloom filter operations." in *AINA*. IEEE Computer Society, 2007, pp. 187–194. [Online]. Available: http://dblp.uni-trier.de/db/conf/aina/aina2007.html#MichaelNPS07

[95] S. Ramesh, O. Papapetrou, and W. Siberski, "Optimizing distributed joins with bloom filters," in *Proceedings of the 5th International Conference on Distributed Computing and Internet Technology*, ser. ICDCIT '08. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 145–156. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-89737-8_15

[96] D. Himali and S. Prasad, "Spun: A p2p probabilistic search algorithm based on successful paths in unstructured networks," in *Parallel and Distributed Processing Work-*

shops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on, 2011, pp. 1610–1617.

[97] L. A. Adamic, R. M. Lukose, A. R. Puniyani, and B. A. Huberman, "Search in power-law networks," *Physical Review E*, vol. 64, no. 4, pp. 046 135+, Sep. 2001. [Online]. Available: http://dx.doi.org/10.1103/physreve.64.046135

[98] A. Song, J. Zhou, and J. Luo, "Cooperation based p2p topology adaptation scheme towards improving search performance," in *Chinagrid Conference (ChinaGrid), 2011 Sixth Annual*, 2011, pp. 147–154.

[99] M. Atencia, J. Euzenat, G. Pirrò, and M.-C. Rousset, "Alignment-based trust for resource finding in semantic p2p networks," in *Proceedings of the 10th international conference on The semantic web - Volume Part I*, ser. ISWC'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 51–66. [Online]. Available: http://dl.acm.org/citation.cfm?id=2063016.2063021

[100] S. Banerjee and T. Pedersen, "An adapted lesk algorithm for word sense disambiguation using wordnet," in *Proceedings of the Third International Conference on Computational Linguistics and Intelligent Text Processing*, ser. CICLing '02. London, UK, UK: Springer-Verlag, 2002, pp. 136–145. [Online]. Available: http://dl.acm.org/citation.cfm?id=647344.724142

[101] E. Cohen and S. Shenker, "Replication strategies in unstructured peer-to-peer networks," in *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM '02. New York, NY, USA: ACM, 2002, pp. 177–190. [Online]. Available: http://doi.acm.org/10.1145/633025.633043

[102] P. Reynolds and A. Vahdat, "Efficient peer-to-peer keyword searching," in *Middleware 2003*, ser. Lecture Notes in Computer Science, M. Endler and D. Schmidt,

Eds. Springer Berlin Heidelberg, 2003, vol. 2672, pp. 21–40. [Online]. Available: http://dx.doi.org/10.1007/3-540-44892-6_2

[103] F. Xue, G. Feng, and Y. Zhang, "Commusearch: Small-world based semantic search architecture in p2p networks," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, 2010, pp. 1–5.

[104] R. Ferreira, M. Krishna Ramanathan, A. Awan, A. Grama, and S. Jagannathan, "Search with probabilistic guarantees in unstructured peer-to-peer networks," in *Peer-to-Peer Computing, 2005. P2P 2005. Fifth IEEE International Conference on*, 2005, pp. 165–172.

[105] G. Gao, R. Li, K. Wen, and X. Gu, "Proactive replication for rare objects in unstructured peer-to-peer networks," *Journal of Network and Computer Applications*, vol. 35, no. 1, pp. 85 – 96, 2012, ¡ce:title¿Collaborative Computing and Applications¡/ce:title¿. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804511000452

[106] P. Haase, R. Siebes, and F. V. Harmelen, "F.: Peer selection in peer-to-peer networks with semantic topologies," in *In: International Conference on Semantics of a Networked World: Semantics for Grid Databases*, 2004.

[107] H. Jin and H. Chen, "Semrex: Efficient search in a semantic overlay for literature retrieval," *Future Gener. Comput. Syst.*, vol. 24, no. 6, pp. 475–488, Jun. 2008. [Online]. Available: http://dx.doi.org/10.1016/j.future.2007.07.008

[108] D. N. Kanellopoulos and A. A. Panagopoulos, "Exploiting tourism destinations' knowledge in an rdf-based p2p network," *J. Netw. Comput. Appl.*, vol. 31, no. 2, pp. 179–200, Apr. 2008. [Online]. Available: http://dx.doi.org/10.1016/j.jnca.2006.03.003

[109] A.-M. Kermarrec and M. van Steen, "Gossiping in distributed systems," *SIGOPS Oper. Syst. Rev.*, vol. 41, no. 5, pp. 2–7, Oct. 2007.

[110] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmér, and T. Risch, "Edutella: a p2p networking infrastructure based on rdf," in *Proceedings of the 11th international conference on World Wide Web*, ser. WWW '02. New York, NY, USA: ACM, 2002, pp. 604–615. [Online]. Available: http://doi.acm.org/10.1145/511446.511525

[111] K. P. N. Puttaswamy, A. Sala, and B. Y. Zhao, "Searching for rare objects using index replication." in *INFOCOM*. IEEE, 2008, pp. 1723–1731. [Online]. Available: http://dblp.uni-trier.de/db/conf/infocom/infocom2008.html#PuttaswamySZ08

[112] M. Ripeanu, "[15] peer-to-peer architecture case study: Gnutella network," in *Proceedings of the First International Conference on Peer-to-Peer Computing*, ser. P2P '01. Washington, DC, USA: IEEE Computer Society, 2001, pp. 99–. [Online]. Available: http://dl.acm.org/citation.cfm?id=882470.883281

[113] I. Rudomilov and I. Jelinek, "Semantic p2p search engine," in *Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on*, 2011, pp. 991–995.

[114] H. Shen and Y. Zhu, "A proactive low-overhead file replication scheme for structured p2p content delivery networks," *Journal of Parallel and Distributed Computing*, vol. 69, no. 5, pp. 429 – 440, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0743731509000264

[115] W. Song, R. Li, Z. Lu, and M. Sarem, "Semsearch: a scalable semantic searching algorithm for unstructured p2p network," in *Proceedings of the 10th Asia-Pacific web conference on Progress in WWW research and development*, ser. APWeb'08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 631–636. [Online]. Available: http://dl.acm.org/citation.cfm?id=1791734.1791808

[116] S. Thampi and K. Sekaran, "Autonomous data replication using q-learning for unstructured p2p networks," in *Network Computing and Applications, 2007. NCA 2007. Sixth IEEE International Symposium on*, 2007, pp. 311–317.

[117] Y.-f. B. Wu, Q. Li, R. S. Bot, and X. Chen, "Domain-specific keyphrase extraction," in *Proceedings of the 14th ACM international conference on Information and knowledge management*, ser. CIKM '05. New York, NY, USA: ACM, 2005, pp. 283–284. [Online]. Available: http://doi.acm.org/10.1145/1099554.1099628

[118] X. Liao, F. Zhang, H. Jin, and L. Yu, "idare: Proactive data replication mechanism for p2p vod system," in *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*, 2010, pp. 682–689.

[119] Z. Li and G. Xie, "rsearch: Ring-based semantic overlay for efficient recall-guaranteed search in p2p networks," in *Parallel Processing Workshops (ICPPW), 2010 39th International Conference on*, 2010, pp. 354–361.