Assisted Peer-to-Peer Search with Partial Indexing

Rongmei Zhang, *Member*, *IEEE*, and Y. Charlie Hu, *Senior Member*, *IEEE*

Abstract—In the past few years, peer-to-peer (P2P) networks have become a promising paradigm for building a wide variety of distributed systems and applications. The most popular P2P application till today is file sharing, e.g., Gnutella, Kazza, etc. These systems are usually referred to as unstructured, and search in unstructured P2P networks usually involves flooding or random walking. On the other hand, in structured P2P networks (DHTs), search is usually performed by looking up a distributed inverted index. The efficiency of the search mechanism is the key to the scalability of a P2P content sharing system. So far, neither unstructured nor structured P2P networks alone can solve the search problem in a satisfactory way. In this paper, we propose to combine the strengths of both unstructured and structured P2P networks to achieve more efficient search. Specifically, we propose to enhance search in unstructured P2P overlay networks by building a partial index of shared data using a structured P2P network. The index maintains two types of information: the top interests of peers and globally unpopular data, both characterized by data properties. The proposed search protocol, *assisted search with partial indexing*, makes use of the index to improve search in three ways: First, the index assists peers to find other peers with similar interests and the unstructured search overlay is formed to reflect peer interests. Second, the index also provides search hints for those data difficult to locate by exploring peer interest locality, and these hints can be used for second-chance search. Third, the index helps to locate unpopular data items. Experiments based on a P2P file sharing trace show that the assisted search with a lightweight partial indexing service can significantly improve the success rate in locating data than Gnutella and a hit-rate-based protocol in unstructured P2P systems, while incurring low search latency and overheads.

Index Terms—Distributed systems, information search and retrieval.

1 Introduction

In the past few years, peer-to-peer (P2P) overlay networks have become popular. The most prevalent P2P application till today is file sharing. P2P file sharing systems have been constantly evolving, for example, from the pioneer Napster to Gnutella, to Kazza, etc. One major driving force behind this evolution is to strive for better search mechanisms. The earliest search (i.e., Napster) was performed by a centralized server. This approach is obviously not scalable. The search protocol widely in use today, e.g., in Gnutella and Kazza, involves flooding the P2P network. Since there is no explicit control over the network topology or data placement, this type of P2P networks are usually referred to as "unstructured." The unstructured network topology is highly robust to failures or node transience. It is also relatively straightforward to implement multiple-keyword search or partial match. On the down side, flooding causes high volumes of network traffic overheads, and the search results are nondeterministic. The performance of search in unstructured P2P overlay networks can be improved by using smarter search or data replication algorithms [7], [35], [40], [8]. Other recent work [6], [33], [41], [4] exploits the inherent locality in peer interests or the heterogeneity in peer capacity.

Meanwhile, a new class of P2P systems have been proposed [24], [34], [30], [43]. This type of P2P systems effectively implement Distributed Hash Tables (DHTs), and are therefore called "structured." In a structured P2P overlay network, search is performed by looking up the DHT. Compared with an unstructured overlay network, routing in a DHT only involves a small number of nodes and completes in a small number of overlay hops. A DHT provides a natural platform for conducting exact-match search: e.g., the hash value of the filename can be used to index and to lookup a file. While it is not as straightforward as in unstructured P2P networks, recently there have been proposals for implementing multiple-keyword search [26] or semantic-based search [37], [36], [17] using DHTs.

One major aspect to distinguish P2P search mechanisms is whether indexing of shared data is employed. In the case of unstructured P2P networks, the earliest system, Napster, maintained a full index of shared files in a centralized server. The now popular Gnutella-like networks do not implement global indexing and instead search in these networks relies on flooding. In P2P networks with super-peers such as Kazza, more resourceful peers maintain an index of the data at nearby peers. In structured P2P systems, inverted lists of documents are published to the DHT under single terms [26], [36] or term (semantic) vectors [37]. Search is conducted by looking up the DHT-based index. In summary, if indexing is used, the index contains information of data items, and such information is directly used for the purpose of resolving queries, or directing queries toward where they are more likely to be resolved.

In this paper, we propose a DHT-based partial indexing scheme. The index maintains the data sharing interests of peers, which can be characterized by their representative data items, as well as information of unpopular data items. The partial index has three complementary purposes.

First, it helps peers to find others with the same interests, and peers are connected to each other based on shared

• *The authors are with the School of Electrical and Computer Engineering, Purdue University, 1285 Northwestern Ave., West Lafayette, IN 47907. E-mail: {rongmei, ychu}@purdue.edu.*

Manuscript received 3 Apr. 2005; revised 16 Mar. 2006; accepted 27 July 2006; published online 9 Jan. 2007.
Recommended for acceptance by J. Fortes.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-0223-0405.
Digital Object Identifier no. 10.1109/TPDS.2007.1035.

interests to form an unstructured overlay where searching is conducted. Due to the interest locality, a query is more likely to be satisfied by nearby peers. Interest locality has been exploited in previous work [6], [33], but only in an implicit way. In [33], a peer discovers potential shared interests with other peers by observing the query history: those with the highest hit-rates are chosen as shortcuts for future queries. A shortcut is only determined after a significant number of queries are received and resolved by the same peer. This passive way of discovering shared interests is agnostic of the actual semantics of the data being shared by peers. The discovery process based on the query history is slow and usually unable to identify all the other peers sharing the same interests that exist in the network. In the assisted search protocol, instead of accumulating the knowledge about other peers' interests from incremental search experience, peers can directly communicate through the index. Specifically, peers register their major interests (e.g., determined from data downloaded in the past) with the index, and also query the index for other peers with the same interests to select as neighbors.

Second, the index also provides search hints for those data difficult to locate based on the interest-based overlay. We expect that the majority of queries can be satisfied by looking up the unstructured search overlay created based on peer interests. For those queries that cannot be resolved, the index can be queried for search hints. Specifically, the index returns pointers to peers that have registered some of the properties being searched for as their top interests. These possible destinations are used for second-chance search.

Finally, the index helps to improve the chances of finding unpopular data. The success of search is closely related to data popularity. In existing unstructured P2P overlays like Gnutella, a popular data item is more likely to be located since there are more replicas in the network, whereas an unpopular data item can not be found unless a large number or all of the peers are searched, e.g., using expanding ring search with increasing TTLs. In this paper, we explicitly address data popularity. Specifically, peers locally monitor the popularity of their own data, and unpopular data, in the form of data properties, are also registered with the index. During search, if the first attempt fails to satisfy the query and the index is contacted for search hints, popularity-based registrations are returned together with interest-based registrations as possible locations for second-chance search.

In summary, the index only maintains information about the top interests of peers and unpopular data. The unstructured search overlay is formed in a way to reflect peer interests through assistance from the index; the search process also benefits directly from the index by retrieving search hints for hard-to-locate data items. We call our proposed search protocol *assisted search with partial indexing*.

Search inside the interest-based unstructured overlay can be performed by flooding, as in the original Gnutella. It can benefit from applying more sophisticated search techniques [19], [40] or allowing caching [7] of search results at intermediate nodes. However, we expect to achieve higher success rate with equal (or even smaller) search scope (e.g., the same TTL value). If the first search attempt fails to locate the data, we optionally resort to the index for search hints. If the hints returned by the index are accurate enough, search overhead can be significantly reduced because search with larger scope, e.g., flooding with larger TTLs, is avoided.

Compared to search in pure structured P2P overlays, the assisted search has inherent support for multiple-keyword search or partial match. Peers query the index overlay only using its few top interests and most queries can be resolved inside the interest-based search overlay without further involving the index. In contrast, in structured P2P overlays, the index must be queried using at least one keyword for each search.

The implementation of the indexing service is separate from the unstructured search overlay. In this paper, the index is maintained by a structured P2P overlay, so that the index information can be retrieved quickly with small overhead. The structured overlay can be constructed on top of the same nodes from the unstructured overlay. Recent studies [32], [20], [21] have shown that a structured overlay can be designed to handle node churns observed in P2P file sharing systems.

The rest of the paper is organized as follows: Section 2 discusses related work. Section 3 presents the design of our proposed assisted search protocol. Section 4 discusses the design of our trace-driven experiments and the evaluation results are presented in Section 5. Section 6 draws conclusions.

2 Related Work

In this section, we discuss previous work related to search in P2P overlay networks.

2.1 Search in Unstructured P2P Overlays

Since the introduction of P2P file sharing systems, P2P traffic on the Internet has grown rapidly. There have been continuous efforts to improve the flooding-based search algorithm. In the random walk algorithm [19], simulation results show the random walk algorithm can significantly reduce the search traffic with slightly increased search latency, compared to the flooding method. Several strategies aimed at improving search efficiency are studied in [40], including expanding ring search and directed random walks. In [5], hybrid search schemes that combines flooding and random walks are studied. The assisted search algorithm presented in this paper can also benefit from applying these techniques. For example, queries can be forwarded to a selected subset of neighbors only.

In [40], each node maintains an index over the data of all nodes within a small number r hops of itself, so that it can process queries on behalf of these nodes. In [8], each node maintains an approximate index of the data available through each neighbor, and such indices can direct queries toward where they are more likely to be satisfied. In contrast to these local indexing schemes, our proposed assisted search protocol maintains a partial index using a logically separate DHT overlay.

The locality embedded in human interests has been recognized for its effectiveness in guiding search queries [6], [33]. In [13], peers with large data repositories self-organize into a cluster, where a query is first forwarded. In addition, a peer also prefetches the indices of other peers that

have the highest hit-rates for past queries, to be used to resolve future queries locally. In the assisted search protocol, instead of gradually learning from history, peers express their interests explicitly and seek others with similar interests actively via the intermediate index overlay. In [14], the authors study the pattern and properties of file sharing between peers with common interests using "data sharing graphs."

SON [9] exploits explicit data semantics and peer interests: peers that are semantically related connect to each other to form a semantic overlay network. However, the semantic classification of data objects, peers and queries is performed by specialized servers, based on a predefined semantic classification hierarchy. The partial index proposed in this paper is distributed and peer interests are identified locally.

Peers are usually treated as equal entities. However, significant heterogeneity may exist in many P2P systems. This heterogeneity has been exploited to improve search performance in terms of throughput and scalability [4]. In the super-peer structure [41], peers of high capacity maintain an index over the data of low-capacity peers called leaf nodes, and resolve queries on behalf of these leaf nodes. In [31], a percolation algorithm is proposed for search in random networks with power-law and heavy-tailed degree distribution. By leveraging high-degree nodes in the network, the percolation search algorithm can locate any content reliably and quickly (in time $O(log\ N)$), while the total traffic scales sublinearly with the network size. These efforts are orthogonal to our main ideas and we do not elaborate on issues regarding peer heterogeneity in this paper.

Search efficiency can be improved by explicitly controlling the replication of data items based on their popularity [19], [7]; the square-root replication is shown to be theoretically optimal in minimizing the search traffic. Alternative data placement approaches and their impacts on search performance are also studied in [35]. In [23], the authors propose transparent caching of query results at organization gateways. In [39], query results are cached at selected nodes based on hashing of the query; similarly, a query is forwarded to selected nodes based on its hashing value. This selective caching and adaptive search protocol is shown to significantly reduce network search traffic. In this paper, we focus on the search algorithm; the data is replicated by the querier upon a successful search, and we do not assume any caching strategies at intermediate nodes.

2.2 Search in Structured P2P Overlays

While much of the effort has been directed toward better search algorithms in unstructured P2P overlays, various applications and services have been built based on structured P2P overlays, such as archival storage systems [10], [29], [15], [22] and group communication mechanisms [3], [45], [25], [42], [2]. The feasibility of providing full-text Web search using P2P networks is studied in [16]. The authors pointed out that the P2P network is not likely to have the capacity of supporting full-text Web search by using existing unstructured or structured search techniques. The assisted search protocol proposed in this paper is an effort toward more powerful algorithms that have the potential of supporting large-scale content search. Only recently, there have been proposals to implement more sophisticated search capabilities such as multikeyword search in structured overlays [26]. pSearch [38], [37] implements full-text search based on CAN [24]: Both data and queries are represented by term (semantic) vectors and search is performed through matching in a multidimensional Cartesian space. eSearch [36] proposes a hybrid global-local indexing scheme for full-text search; documents together with their full term lists are published only under the top-ranking keywords. In Semantic Small World (SSW) [17], peers self-organize into a small-world overlay network based on the semantics of their local data, and peers sharing similar data objects form clusters. Although the semantic space is of a high dimension, like in pSearch, peers reside in a one-dimensional space through dimension reduction. In [44], the authors propose a super-peer based lookup algorithm for heterogeneous structured P2P system. High-capacity peers form a super-peer DHT as an enhancement to the original DHT, and such super peers act as local servers for less resourceful peers.

2.3 Search in Hybrid P2P Overlays

In Structella [1], the unstructured overlay in Gnutella is replaced with a structured overlay. By taking advantage of structure, this hybrid system eliminates redundant search traffic from the flooding or random walk search algorithms.

In probabilistic location [27], a lossy distributed index (i.e., implemented by attenuated Bloom filters) is used to locate data close to the querier; the default DHT-based algorithm is invoked if the probabilistic location fails. While probabilistic location provides an enhancement to structured P2P search systems, our approach is opposite in that search is performed in the unstructured P2P overlay with the assistance from a structured index overlay.

In Yappers [12], both data and peers are mapped to a small number of buckets, and each peer builds a small DHT of nearby neighbors based on this mapping. Both publishing and querying are guided by the mapping so that only the peers that fall in the same bucket as the data are involved. Such binning of data and peers is "blind" (by hashing data/peer IDs) and does not consider data content.

Very recently, the work in [18] was brought to our attention. In [18], the impact of data popularity on search results is measured and a hybrid of unstructured and structured search scheme is proposed. The assisted search protocol approaches the problem of searching unpopular data from two complementary directions: exploring locality in peer interests through interest-based clustering and selective indexing of unpopular data.

2.4 A Taxonomy of P2P Indexing Schemes

Indexing is a fundamental component in P2P search schemes. To aid in the understanding of our assisted P2P search protocol and the difference from other P2P search mechanisms that have been proposed so far, we provide a taxonomy of P2P search according to how indexing is applied. In particular, we classify P2P indexing and search schemes along two orthogonal dimensions. The first dimension characterizes the *scale* at which indexing is generated, i.e., whether indexing involves only nodes that reside within a local neighborhood or all nodes in the network. The second dimension characterizes the *extent* to which indexing is performed, i.e., whether the index is designed to encompass all data items in the network.

TABLE 1
A Taxonomy of Indexing in P2P Search Mechanisms

	Local-scale	Global-scale
Partial	routing indices [8]	assisted search (this paper)
Total	super-peer [41], [44], probabilistic location [27]	pSearch [37], eSearch [36], and [26]

Table 1 summarizes representative work in each category along these two dimensions. In super-peer networks, including both unstructured [41] and structured [44] networks, the indexing is local-scale, but every data item in the network is indexed; therefore, it is local-scale full indexing. The probabilistic location approach [27] belongs to the same category, although Bloom filters are used for indexing data in local-area nodes. Routing indices summarize the amount of data available through each neighbor within a small number of hops; in addition, routing indices also include the number of documents on selected topics. Hence, routing indices provide local-scale coarse-grained partial indexing. DHT-based indexing schemes that we have discussed in this section [37], [36], [26] are global-scale full indexing. In contrast to these indexing schemes, the partial index that we propose in this paper maintains the top interests of individual peers, as well as pointers to rare data items, and it is implemented by a global-scale DHT. Therefore, it is distinct from previously proposed indexing schemes as a global-scale partial indexing scheme.

3 Design

In this section, we discuss the design of the assisted search protocol with partial indexing, including how the partial index is maintained and how this partial index is utilized to improve the search performance.

3.1 Definitions

Before describing the assisted search protocol with partial indexing, we first introduce several key concepts used throughout this paper.

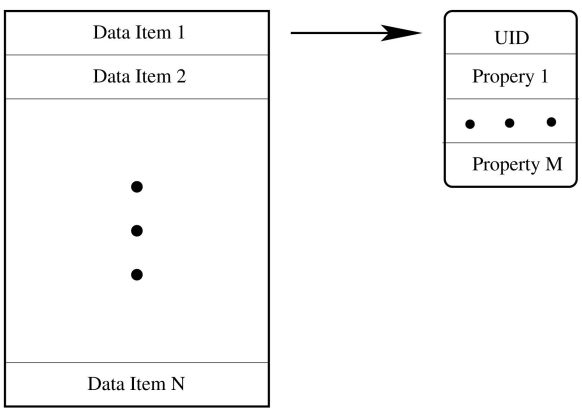


Fig. 1. Data possessed by a peer. Each data item contains one or more properties.

Peers can uniquely identify the data items that they possess, e.g., by file indices in Gnutella. A data item is also associated with a number of *properties* (keywords), e.g., the properties of a song may include the title, the singer, and the musical genre, etc. These metadata are used in specifying queries. Fig. 1 shows the data possessed by a peer and the properties contained in each data item. By defining "properties" as the top ranking keywords from a document, the protocol presented here is also applicable to content-based full-text search.

A peer's local *interests* are with respect to the peer itself, and defined as the most frequent properties contained in the queries that it issues. A peer's future interests are approximated with its past interests, which in turn are approximated with the properties of the data that it currently possesses. In other words, the interests of a peer are represented by the dominant properties of its data possession. Fig. 2 shows the local interests of the peer in Fig. 1.

An important factor affecting the success of search is the *popularity of data*. Different from a peer's local interests, the popularity of a data item is with respect to all peers in the search network, and is defined by its availability, i.e., the more popular the data, the more replicas are available and the easier it is to locate a replica. Under the assumption that at least one of the query hits is replicated by the querier after a successful search, the frequency at which the data item is queried is a good estimation of its popularity. Since queries are specified by properties of the data being searched for, we consider the *popularities of properties* being searched as opposed to the popularities of data items themselves in the rest of this paper. Specifically, data

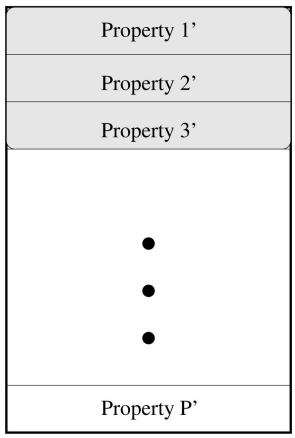


Fig. 2. Merged properties of local data items, sorted according to the frequency of appearance. Top ranked properties (highlighted) become the local interests of the peer.

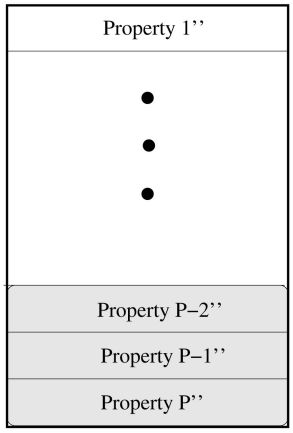


Fig. 3. Merged properties of local data items, sorted according to the frequency of appearance in received queries. Bottom ranked properties (highlighted) that are not part of local interests become the locally-observed unpopular properties.

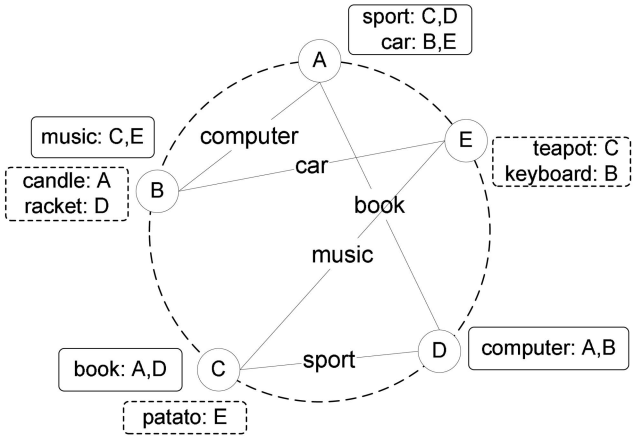


Fig. 4. Example of an assisted search network. The dashed circle represents the structured index overlay and the solid lines represent links between the five nodes in the unstructured search overlay. Nodes participate in both the index overlay and the search overlay. The search overlay is formed based on shared interests between nodes. The solid box and the dashed box next to each node contain registered interests and unpopular properties (of remote nodes) in the index overlay, respectively.

popularity is observed by individual peers and those properties of local data that are seen the least frequently in passing queries are identified as "unpopular" (see Fig. 3).

3.2 Overview

This section gives an overview of the assisted P2P search protocol with partial indexing. The P2P network consists of two logical overlays. Search is performed in an unstructured overlay which forms the "search" overlay, and the index is implemented as a structured overlay which forms the "index" overlay. The distinction between the two components is purely logical; in this paper, we assume that each peer participates in both overlays. Fig. 4 shows an example of five nodes A to E. The dashed circle represents the structured index overlay and the solid lines represent links between the five nodes in the unstructured search overlay. The index overlay assists the search overlay to improve its search performance in the following three ways.

First, peers communicate their interests via the index overlay and the search overlay is constructed based on peer interests. Specifically, each peer registers its own major interests, i.e., the most representative properties of local data, with the index overlay and also looks up other existing peers sharing the same interests. In this way, a peer always connects to those peers in the search overlay that share common interests and as a result the propagation of a query tends to first reach those that are more likely to possess the data being searched for. Similar to [6], [33], this approach exploits the locality in human interests. In Fig. 4, each of nodes A, B, C, and D maintains registrations for at least one property (shown within solid boxes). Each node is connected to two other nodes in the search overlay, and each link is labeled with the associated interest property shared by the two end nodes.

Second, neighbors only reflect a peer's top few interests due to limited node degree, although peers are likely to have more diverse interests which cannot be covered by those registered to the index overlay. Thus, a small portion of queries may not be resolved by searching the interest-based overlay alone. In this case, the interest-based registries maintained by the index overlay can be consulted for hints about where to forward such a query for a second try.

Finally, peers also identify the properties from their local data repository that are globally unpopular (from their own observation of passing queries) and are not part of local interests either. These properties represent those data that is difficult to locate by exploring peer interests. They are explicitly registered with the index overlay. Together with interest-based registries, (un)popularity-induced registries can also be returned as potential destinations when the index overlay is queried for search hints. In Fig. 4, nodes B, C, and E also maintain registrations of unpopular data (shown in dashed boxes).

3.3 System Initialization

The bootstrap of the assisted P2P search network involves initializing the search and index overlays. The index overlay is constructed according to the bootstrap mechanism of the corresponding structured P2P overlay. For example, when Pastry [30] is used, the overlay is built as each node joins the network following the joining protocol, i.e., by routing a special message keyed with the new node's identifier. Since peers are supposed to join both overlays, each of them serves as its own entry point into the index overlay, i.e., each peer can access the index overlay directly.

After joining the index overlay, a new peer can obtain the addresses of other peers with the same interests through *lookup(key)* operations in the structured index overlay. For example, a peer can look up "jazz" by setting the key of the query message to be the hash value of "jazz." This query message is received by the index overlay node that maintains pointers to other nodes that are also interested in "jazz." The new peer can choose from these nodes to initialize its neighbor connections.

If a peer cannot determine its interests when first joining the search overlay, it can query the index overlay using random keys and, hence, is connected to the search overlay

through randomly selected neighbors. The connections can then be refined as the peer learns its interests over time.

3.4 Indexing Interest Properties

As a peer accumulates data items by issuing queries and then downloading from other peers, its interests are automatically reflected by its data repository. Top ranking properties of local data are selected as representing interests to be registered with the index overlay.

In ranking local data properties, the age of data items (e.g., when the data items were downloaded) can also be taken into account. For instance, the rank of a data property can be weighted by the age of the corresponding data items such that newly acquired data items contribute more to the overall rank. An alternative is to only count data items below a certain age. Considering the age factor of data exploits the short-term interest locality, as opposed to the long-term interest locality.

The number of registered data properties is determined by peers locally. First the average number of data items per property is computed at each peer, and then every data property with larger than this average number of data items is registered. In addition, the size of a peer's data repository can also be taken into account: If there are many properties associated with large numbers of data items in a peer's local data repository, the peer can register more local properties than others. However, this requires the availability of global-scale information, e.g., the average number of data items per property at each peer.

Peers register their top interests through *insert (key, value)* operations in the structured index overlay. Peer interests can be maintained as soft-state and updated periodically. Alternatively, peers can update the index overlay only when local interests are changed, i.e., by registering new interests and unregistering lost interests, and rely on DHT mapping to maintain the persistence of registered interests. It is optional for peers to provide extra information such as the local number of data items associated with the property, or the local network connection speed.

3.5 Indexing Unpopular Properties

The assisted search protocol also benefits from the registration of rare properties to the index overlay. Like the registration of interests, the decision of registering which (unpopular) properties is made by peers locally. Property popularity can be determined from observing passing traffic. Each time a query message is received, it is searched against the local data cache. Meanwhile, the peer also updates its local record of the search frequency of each property in the query's search criteria. The higher the search frequency, the more popular the property and the more likely that the associated data is well replicated. In particular, first the peer computes the average number of queries that have been received locally for each property; if the number of received queries for a property is below this average, that property can be selected as unpopular. Moreover, the number of replies forwarded back to the querier along the search path is also a good indication of popularity.[1]

An unpopular property as observed locally by a peer is registered only if the peer possesses data items with that property and the property is not already registered as one of local interests. Similar to peer interests, unpopular properties can also be updated periodically or on-demand, and the registries need only contain pointers to the data providers (as opposed to individual data items at the providers). There is a trade-off between the overhead of maintaining unpopular data registries at the index overlay and the ability to search unpopular data. This trade-off can be controlled by the threshold of "unpopularity." For instance, to reduce the number of properties that are registered with the partial index, we can reduce the threshold for selecting unpopular properties by 20 percent.

Similarly as in the selection of peer interests, the freshness of queries can also be taken into account when selecting unpopular properties. For instance, we can consider only queries that are received within a certain time window in the past. Alternatively, queries can be weighted by age so that the properties from recently received queries are given higher priority during the selection.

In order to improve the accuracy of local popularity estimation, the index overly also provides feedbacks regarding data popularity, since it has a global view by accepting registrations from the entire search overlay. If the index receives registrations for a purported unpopular property from more than M peers, this property cannot be counted as "unpopular." Any peer that submits a registration of this property in the future is notified to stop registering it. This allows peers to improve the accuracy of their local popularity estimation and to avoid overloading the index overlay with unnecessary registrations.

3.6 Search Overlay Maintenance

Peers also continuously update their neighbor connections in the search overlay. Periodically, a peer queries the index overlay for other peers sharing the same interests and update their neighbors accordingly. This allows the search overlay to adapt to changing peer interests.

Usually, the index returns more than one candidates for each interest property. The peer can choose from the candidates randomly, or select the best one based on some performance metrics, e.g., the candidate with the largest number of data items for the associated property, or the candidate with the highest network connection speed. The choice is also made according to the neighbor selection scheme used by the peer (as described below).

Each peer determines independently the number of (outgoing) neighbors, e.g., according to the local capacity. The number of neighbors may be orthogonal to the number of registered local interest properties. The association between neighbor peers and local interests can vary. Each neighbor can represent a distinct local interest property. Alternatively, multiple neighbors can be selected to represent the same property of strong local interest. On the other hand, a neighbor can also be associated with more than one local interest properties. For instance, a peer may choose the candidates whose interest list overlaps the most with the local interest list.

Different from Gnutella, the links in the search overlay are uni-directional and each peer is given the maximum flexibility in selecting its own outgoing neighbors. As described above, peers determine independently the number and the selection of neighbors. By decoupling the

1. In Gnutella, query hits are sent along the same paths traveled by the query.

incoming and outgoing connections, peers have minimum interference with each other in neighbor selection. Similarly, each peer determines its own indegree independently.

The frequency of neighbor update should adjust to the evolving speed of peer interests, which is partially reflected by the query rate. Intuitively, the more queries issued, the more downloads made by peers and, thus, the more quickly peer interests change. There is a trade-off between the responsiveness to peer interest changes and the overhead from neighbor updates. In addition, a peer should avoid unnecessary neighbor switches: existing neighbor connections should be preserved if they still satisfy the neighbor selection criteria.

We assume that a peer presents its own interests to the index overlay in an honest way. A malicious peer might misguide other peers by registering false interests. Such misdeed can be detected by monitoring the actual hit rate of neighbors and dropping those connections that cannot fulfill the expected performance.

3.7 Resolving Queries

A query is first issued to the search overlay. For example, if controlled flooding such as the Gnutella search protocol is used, the query is forwarded on to neighbors until the TTL value reaches zero. If random walking is used, the query is only forwarded to a random subset of neighbors at each step. Alternatively, we can use biased walking by forwarding the query message to the neighbors whose interests overlap the most with the query string. If the first try in the search overlay yields no hits at all or the peer is not satisfied with the results, e.g., not enough hits are generated, the peer has a second chance by seeking search guidance from the index overlay, i.e., the index overlay is queried for nodes that are likely to satisfy the search. One or more properties from the search criteria can be used. The destinations returned by the index overlay include both the peers that have strong interests in the corresponding properties and the peers that have registered the properties as unpopular. The querier then contacts these potential destinations sequentially or concurrently in small batches until the search requirement is fulfilled.

Since the index overlay only returns possible destinations, resolving the query still involves searching these destinations to match the entire search criteria, as in the first search attempt. Therefore, although the query into the index overlay for search hints is on a per-property basis, the search itself still retains such appealing features as multi-criterion and partial match.

It is optional for peers to select destinations for the second search attempt based on some performance metrics when extra information regarding the interests or the capacities of the destinations are available. For example, all qualified destinations can be ranked based on their network connection speeds and the top candidates are contacted first.

3.7.1 Alternative Search Algorithm

One variation for the search algorithm is to estimate the possibility of success using the first search attempt (e.g., TTL-controlled flooding in the search overlay) before issuing the query. If the query is unlikely to be resolved by the first attempt, the index overlay can be consulted directly. For instance, if all of the properties being searched for have less than the local average number of data items per property, the index overlay should be contacted for search hints. If no search hints are returned, or the search hints cannot satisfy the query, the peer can also be given a second chance by resorting to the default TTL-controlled flooding. Therefore, this alternative algorithm can achieve the same success rate in resolving queries. However, how the search latency and overhead are affected depends on the accuracy of the estimation.

3.8 Scalability

Since each node participates in both the search overlay and the index overlay, the size and capacity of the index overlay naturally scales when more nodes join the network. Assuming the same query rate and the same amount of local data at each peer, the overhead for maintaining the partial index and for resolving queries will remain at the same level, even as more peers join the network. However, as a peer issues queries and downloads from other peers, the size of its local data repository can increase over time. As a result, the number of interest properties that are submitted to the index overlay can potentially increase over time. To avoid this potential increase, we exploit the observation that a peer's current interests are more accurately reflected by the recently issued queries and, hence, recently downloaded data items. Therefore, when we consider the age of local items in selecting interests (discussed in Section 3.4), e.g., only those items whose age is below some threshold are used in selecting interests, the number of interest properties registered from each node will not increase as the total amount of local data at each peer increases. Similarly, when we consider the age of passing queries in selecting unpopular properties and infer the current popularity of data from queries that are received recently at a node (discussed in Section 3.5), the number of unpopular properties that are registered from each node with the index overlay will not increase over time.

4 Experimental Methodology

We have evaluated the assisted search protocol using trace-driven simulations. In this section, we discuss the design of the experiments.

4.1 Query Workload

We use a trace collected from a real Gnutella file-sharing system for our experiments. The trace was collected using modified Gtk-Gnutella version 0.93, which is a Unix Gnutella client software based on the GIMP Toolkit (GTK+). Changes were made to record Query and QueryHit messages, but the behavior of the client was not affected in any way. Four clients were run in legacy mode, each with minimum and maximum of 10 and 30 neighbors, respectively. The trace was collected over a period of approximately two days (15 March-21 March, 2005). In this period, all passing Query and QueryHit messages were recorded.

The collected data was then processed to extract all queries issued by immediate neighbors of the collecting clients, plus all the corresponding replies. The Gnutella

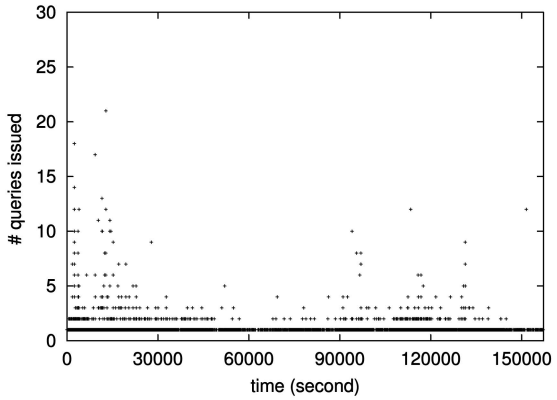


Fig. 5. Query rate: number of queries per second.

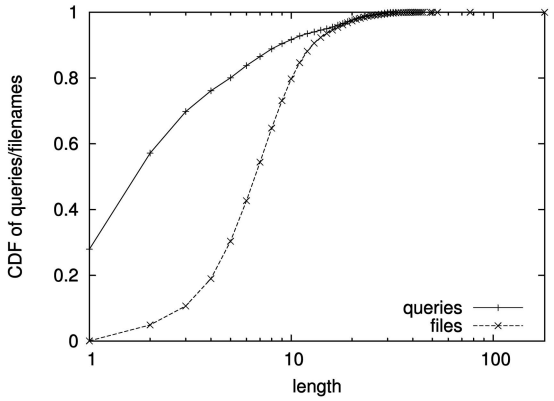


Fig. 6. CDF of query/filename length (number of keywords).

protocol allows the trace collector to identify those queries originated from the immediate neighbors from either the TTL field or the Hops field in the Gnutella packet header. Since many clients use nondefault TTL value, we used the Hops field. We removed the queries without any replies and Fig. 5 shows the Query message rate.

The IP addresses in the trace were anonymized, as well as the Search Criteria field in the Query messages and the filenames in the QueryHit messages. The anonymizing was performed by uniquely mapping each word within the query strings or filenames to its anonymized form. Files (data items) are uniquely identified by their names and the words appearing in a filename are treated as the properties of the associated file. This allows us to simulate multiple properties per data item. The processed trace consists of 1,703 peers, 518,909 files, and 129,363 words (keywords). Fig. 6 shows that the majority (around 95 percent) of queries and filenames contains 15 or fewer keywords.

Fig. 7 shows the frequency of keywords appearing in filenames in the trace: about 57 percent keywords are seen in only one filename and more than 80 percent in at most five filenames. Therefore, a large number of keywords are not likely to be selected as peer interests.

Fig. 8 and Fig. 9 depict the popularity of keywords in the Gnutella trace. First, about 57 percent of keywords are queried by only one peer and almost 75 percent by at most two (Fig. 8). Second, the appearance frequency of keywords in query messages follows a similar distribution (Fig. 9). These statistics suggest that the popularity of keywords is

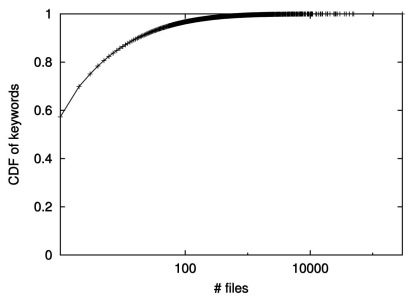


Fig. 7. CDF of keywords versus number of filenames containing the keyword.

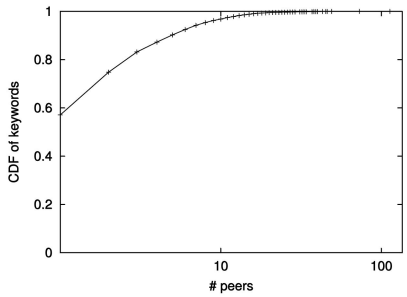


Fig. 8. CDF of keywords versus number of peers querying the keyword.

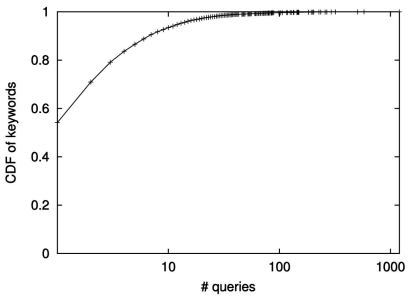


Fig. 9. CDF of keywords versus number of queries containing the keyword.

highly skewed in the Gnutella trace. For example, about 87 percent of keywords are accessed by five or fewer peers.

Before the simulation starts, each file is placed at the node from where it is first seen in a QueryHit message. Since each Query message may yield more than one matching results, the querier randomly selects one file for downloading. We assume that each file is available for sharing from the moment of being replicated.

4.2 Protocol Configurations

In our experiments, we assume that each peer in the trace participates in both the index overlay and the search overlay; each overlay consists of 1,703 nodes. The index overlay is a Pastry (FreePastry [11]) network and peers join it with random node identifiers. We have developed a packet-level simulator to evaluate the search protocols and configurations are described in the following.[2]

The assisted search protocol is configured in several ways in order to study the contributions of each component of the protocol. In the first configuration, the search network is constructed and maintained based on

2. The simulator is available from the authors.

peer interests (labeled "interest-based"). In the second configuration, a query is given a second-chance if it cannot be satisfied by the first search attempt in the search overlay. The index overlay is contacted for search hints, and the addresses of peers whose registered interests match the search criteria are returned as the potential destinations for the second try (labeled "interest-based $\times$ 2"). In the the third configuration, pointers to unpopular data properties are also returned by the index overlay as equally possible locations for the second search attempt (labeled "$\mathrm{interest} + \mathrm{popularity} - \mathrm{based} \times 2$").

In the search overlay, we assume uniform outgoing connectivity for all peers. Each peer is connected to four neighbors, which is close to the average degree of a Gnutella network [28]. We do not consider heterogeneity in peer capacity and do not bound peer indegree in our experiments. Initially, peers are connected in a random manner, i.e., neighbors are selected randomly. As the simulation continues, each peer updates its neighbors based on local interests, as described in Section 3.6. Specifically, each of the four neighbors represents one distinct interest property (keyword). The interval for selecting local interests and unpopular data properties is set to be 10 minutes; in this experiment, we use on-demand registration and the index overlay is updated only when local selection of interests or unpopular properties is changed during each interval. The interval for updating neighbors based on local interests is 30 minutes. The search overlay implements the Gnutella protocol, e.g., a query is forwarded to all outgoing neighbors except the one from which the query comes from. The flooding of queries is controlled by a TTL. The default TTL is set to be 4.

In the simulation, a property that has more than the average number of local data items per property is registered with the index overlay as part of local interests. A property is considered unpopular if the number of associated queries is less than the average number of locally recorded queries per property. Such a property is only registered if it corresponds to at least one data item in the local data repository and does not overlap with registered local interests. We do not consider aging of peer interests or data popularity in our simulations. The index overlay also provides feedbacks to the search overlay regarding data unpopularity observations. Specifically, if an indexing node sees 10 or more registrations for a particular property, it labels this property not qualified as unpopular.

We compare the three versions of the assisted search protocol with the Gnutella protocol (labeled "Gnutella"), where the search overlay is formed randomly and flooding is used for resolving queries. We also compare the "interest-based" version with a "history-based" counterpart. The "history-based" scheme only involves the search overlay and updates neighbors in a way similar to how the shortcuts are selected in [33]; it updates peer connections based on learned history, instead of explicitly expressed local interests. Specifically, each peer keeps track of the hit rate of its queries at other peers and chooses the top ranking peers as neighbors. The interval of updating neighbor connections is set to be the same as in the "interest-based" scheme.

4.3 Performance Metrics

We use the following metrics to compare and analyze the performance of the evaluated search algorithms:

1. *Success Rate:* The success rate measures the effectiveness of the search algorithm and is defined as the average percentage of queries that are resolved successfully. A query is resolved successfully if at least one reply is received, either from the first or the second search attempt. A file is considered a match for a query only if the filename contains all the keywords in the query string. We do not consider partial match in our experiments.
2. *Search Delay:* The search delay is measured by the average time elapsed until the first reply is received at the querier and is only defined for successfully resolved queries, e.g., queries that generate replies matching all the query keywords.
3. *Overhead:* We measure the overhead of each search algorithm using the average bandwidth consumption at each node. For the assisted search scheme, this includes the overhead from maintaining and using the partial index. In addition, we also measure the overhead at the index overlay using the average number of registries maintained at each node.
4. *Search Scope:* The search scope is defined by the fraction of peers contacted in resolving a query on average. It characterizes the messaging overhead in the search overlay.
5. *Node Indegree:* Node Indegree measures the impact of the search algorithm on the overlay topology. The distribution of incoming connections at the peers reflects the degree of clustering by the interest-based or history-based search algorithms.

5 Experiment Results

This section presents the simulation results of evaluating the performance of the assisted search protocol.

5.1 Success Rate

Fig. 10 shows the success rate of search using the Gnutella trace. To better understand the potentials and limitations of our proposed search protocol, we introduce the "optimal" success rate, which is the success rate if queries are flooded to the entire network. The optimal success rate is between about 84-92 percent, as shown in Fig. 10. During processing the Gnutella trace, we have observed that for some queries, there exist a number of replies that do not match all the keywords in the query string. This is probably because in some Gnutella servants, other metadata than filenames are used for matching during search. Although the network size is fixed at 1,703 nodes throughout the simulation, the result fluctuates slightly as the queries are issued and each query may or may not be resolved successully.

The search success rate from the Gnutella protocol is around 60 percent. The "history-based" scheme yields slightly better (approximately 2 percent) result, while the "interest-based" scheme achieves 5-7 percent higher success rate than the Gnutella protocol. By giving a second chance to those queries that cannot be fulfilled by the first search attempts, the success rate of the "interest-based" scheme can be further improved by 20 percent. In

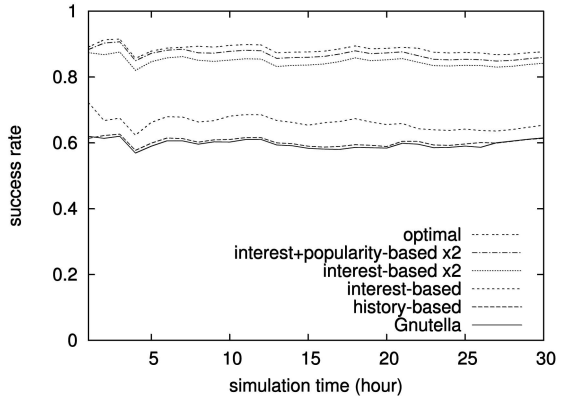


Fig. 10. Success rate of search. The "interest-based" scheme is more effective than the "history-based" scheme. With second-chance search, the assisted search protocol achieves close to optimal success rate.

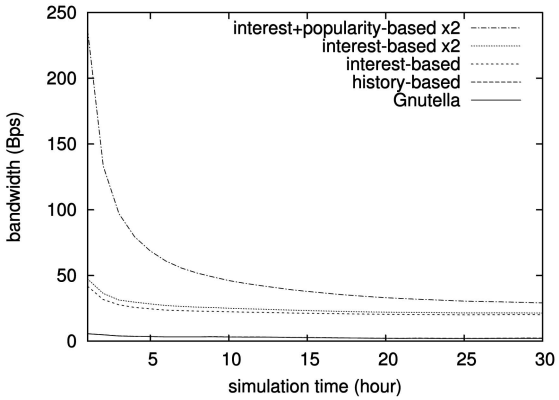


Fig. 12. Bandwidth overhead at each node. The partial index incurs small bandwidth overhead.

fact, the results from both the "interest-based $\times$ 2" and "$\mathrm{interest} + \mathrm{popularity} - \mathrm{based} \times 2$" are very close to the optimal success rate.

5.2 Search Delay

The search delay over time is depicted in Fig. 11. Without using second-chance search, the "interest-based" scheme achieves similar latency to the Gnutella protocol or the "history-based" scheme. When second-chance search is enabled, the average delay to receive a successful reply becomes (about 2.5 hops) higher. This is because the queries satisfied by second search attempts experience longer delay, which includes the timeout ($2 \times$ TTL) for the first search attempt, the delay to retrieve search hints from the index overlay, and the delay to finally resolve the query through the second search attempt. As the "$\mathrm{interest} + \mathrm{popularity} - \mathrm{based} \times 2$" scheme resolves slightly more queries than the "interest-based $\times$ 2" scheme through second-chance search (Fig. 10), it also experiences slightly higher search delay.

5.3 Overhead

Fig. 12 reports the bandwidth overhead of each search algorithm, more specifically, the average bandwidth consumption at each node. For Gnutella and the "history-based" scheme, this overhead comes from the query and query reply messages and, thus, varies with the query rate (Fig. 5). For the assisted search scheme, the bandwidth overhead also consists of the overhead from maintaining and using the partial index: more precisely, the overhead from registering/unregistering interests and unpopular keywords, the overhead from sending popularity feedback to the search overlay, and the overhead from looking up the partial index. We can see that the overhead is highest at the beginning of the experiments as interest and unpopular keywords are first submitted to the index overlay and the curve flattens out later on as only the differences in peer interests or unpopular keywords are updated with the index overlay. Maintaining peer interests at the partial index incurs an average bandwidth overhead of about 20 Bps at the steady state and maintaining unpopular keywords incurs an additional bandwidth overhead of about 200 Bps at the beginning and about 10 Bps toward the end of the simulation. Such overhead is a small price to pay for achieving higher query success rate.

Fig. 13 shows the number of registries maintained by an indexing node in terms of peer interests and unpopular data, respectively. First, we can see that unpopular data registries outnumber peer interest registries, and this can be explained by the large number of rarely seen keywords in the Gnutella trace. Second, the results remain largely fixed throughout the simulation (the number of unpopular data registries decreases slightly), although peers accumulate data items through searching and downloading over the simulation time. This shows that the algorithms used in our experiments for selecting interest properties and for estimating unpopular data properties is adaptable to peer interests and data popularity dynamics.

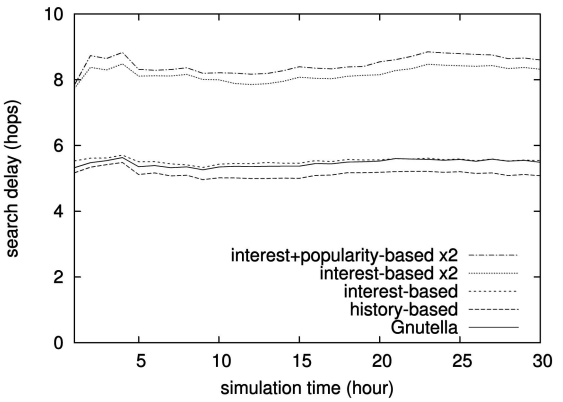


Fig. 11. Search delay. The "interest-based" scheme incurs similar search delay as the baseline scheme. Second-chance search requires additional delay to resolve the query.

5.4 Search Scope and Node Indegree

The search scope remains stable during the simulation, as shown in Fig. 14. For the Gnutella search protocol, each query is received by about 17 percent of all peers. The search scope of the "interest-based" scheme and the "history-based" is slightly lower at about 16 percent. Search scope is closely tied to the topology of the search overlay. Although the outdegrees of peers are uniform in our experiments, their indegrees can vary. Fig. 14 implies that the topological features of the search overlay, in terms of node indegree distribution, remain largely unchanged throughout the experiments. Fig. 15 confirms this observation by showing the CDF of node indegree at the end of the

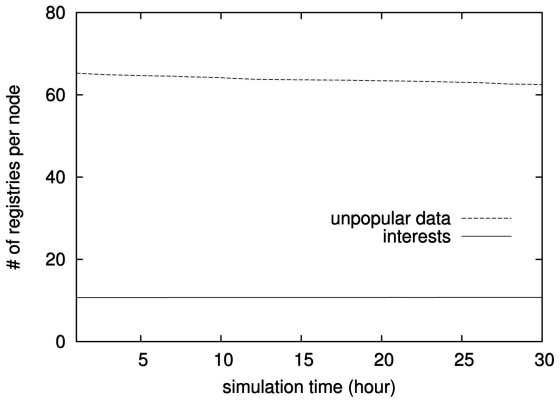


Fig. 13. Average index registries maintained per node. The unpopular keyword registries decrease slightly over time.

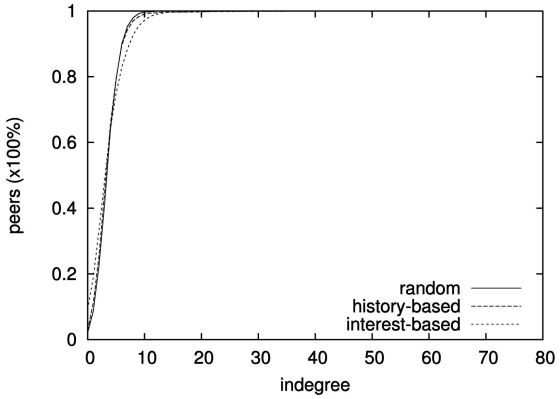


Fig. 15. CDF of node indegree. The "clustering" effect from the "interest-based" or "history-based" schemes is limited.

simulation: Although the CDF of node indegree in the search overlay has a small tail for the "interest-based" or "history-based" scheme, the differences are insignificant when compared with the random overlay topology from the Gnutella search protocol. The maximum node indegree is 13, 34, 63 for the Gnutella, "interest-based," and "history-based" schemes, respectively.

5.5 Discussions

The result shown in Fig. 10 suggests that selecting neighbors based on peer interests alone ("interest-based") does not improve the search success rate significantly. This can explained as follows: In our experiments, each peer has four outgoing connections, and each neighbor represents a distinct interest property. Furthermore, we do not consider partial match and a query is considered successful only if all the keywords in the query string are matched. Together, these two factors contribute to the relatively small gain in the search success rate of the "interest-based" scheme. Nevertheless, the index overlay is very helpful in providing search hints for second-chance search attempts.

From Fig. 10 and Figs. 12 and 13, we can see that although search hints from the registries of unpopular data bring the success rate closer to the optimal, the improvement comes at a relatively considerable indexing overhead. This can be explained by the fact that only a very small fraction of queries are for unpopular data, as a high query frequency naturally leads to a high replication degree for the data being searched for. These queries for unpopular data are not likely to be satisfied from the first search attempt in the interest-based overlay or from the second search attempt based on peer interests. Meanwhile, there exist a large number of unpopular keywords (Figs. 8 and 9) in the trace, which result in a high index maintenance cost. In fact, more than 50 percent of keywords appear in only one query and such keywords are selected as "unpopular" and registered with the partial index although they do not contribute to the search success rate of the "$\mathrm{interest} + \mathrm{popularity} - \mathrm{based} \times 2$" scheme. Despite the potential overhead, maintaining a partial index of unpopular keywords makes it possible to quickly search for unpopular data items without flooding the entire network. In our future work, we will study how to reduce the indexing overhead for unpopular data, e.g., through smarter selection mechanisms for unpopular keywords.

We have also evaluated the effects of expanding the scope of flooding for the Gnutella protocol. Fig. 16 and Fig. 17 show the query success rate and corresponding bandwidth overhead from using $\mathrm{TTL} = 5$ and $\mathrm{TTL} = 6$, compared with the results from using $\mathrm{TTL} = 4$ and the assisted search scheme. Increasing the TTL value by 1 improves the success rate but the result is still below that of the assisted search protocol. Increasing the TTL value by 2 (to $\mathrm{TTL} = 6$) can achieve close to optimal success rate, but the bandwidth overhead is significantly higher in the mean time, and the gap widens over time.

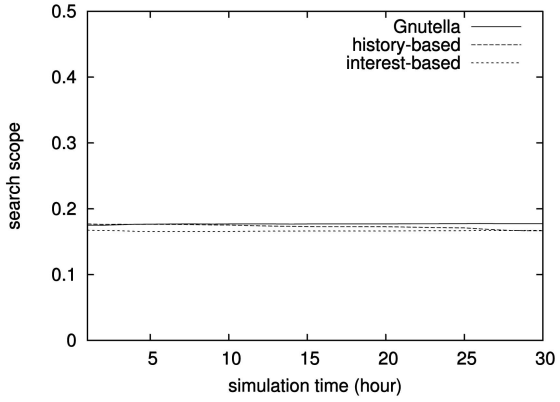


Fig. 14. Search scope. The search scope from the "interest-based" and "history-based" schemes are slightly lower than the random scheme.

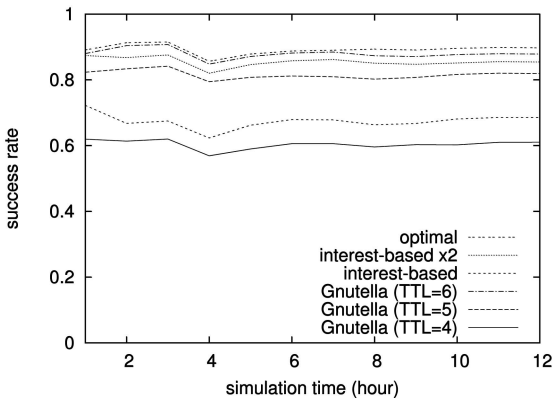


Fig. 16. Success rate of search with increased TTL value.

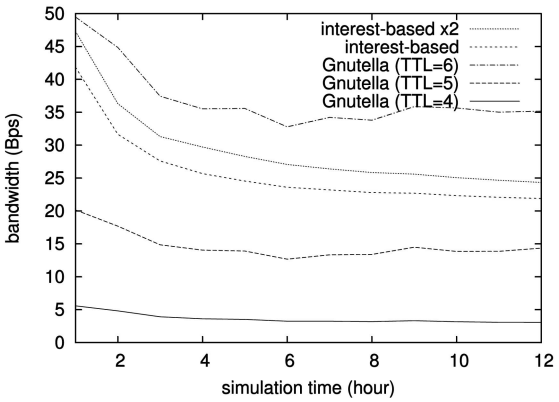


Fig. 17. Bandwidth overhead at each node with increased TTL value.

6 Conclusions

In this paper, we have presented a new protocol for P2P search with the assistance from a partial indexing service based on peer interests and data popularity. The assisted search protocol leverages the advantages of both unstructured and structured P2P systems. The partial index is built on top of a structured P2P overlay network, and it maintains the top interests of peers and pointers to globally unpopular data. Peers can locate others with similar interests through quick lookup operations into the index overlay. The search overlay has an unstructured topology and peers have the flexibility to select their neighbors based on local interests. The partial index also helps to search for data difficult to locate by traversing the interest-based search overlay; search hints in the form of pointers to likely locations can be retrieved for second-chance search. In summary, the assisted search protocol exploits the locality between individual peer interests and meanwhile recognizes global data popularity. With the assistance from the structured partial index, it achieves higher search efficiency than a pure flooding-based or history-based search scheme. At the same time it also retains desirable features of search in unstructured overlays such as versatileness and robustness. Experiments based on a trace from a real P2P file sharing application show that the assisted search with partial indexing is a promising solution to the search problem in P2P content sharing systems.

Acknowledgments

This work was supported in part by US National Science Foundation (NSF) CAREER award grant ACI-0238379.

References

[1] M. Castro, M. Costa, and A. Rowstron, "Should We Build Gnutella on a Structured Overlay?" *Proc. ACM HotNets,* Nov. 2003.

[2] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-Bandwidth Content Distribution in Cooperative Environments," *Proc. ACM Symp. Operating Systems Principles (SOSP),* Oct. 2003.

[3] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, "Scribe: A Large-Scale and Decentralized Application-Level Multicast Infrastructure," *IEEE J. Selected Areas in Comm.,* Oct. 2002.

[4] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making Gnutella-Like P2P Systems Scalable," *Proc. ACM SIGCOMM,* Aug. 2003.

[5] C. Gkantsidis, M. Mihail, and A. Saberi, "Hybrid Search Schemes for Unstructured Peer-to-Peer Networks," *Proc. IEEE INFOCOM,* Mar. 2005.

[6] E. Cohen, A. Fiat, and H. Kaplan, "Associative Search in Peer-to-Peer Networks: Harnessing Latent Semantics," *Proc. IEEE INFOCOM,* Apr. 2003.

[7] E. Cohen and S. Shenker, "Replication Strategies in Unstructured Peer-to-Peer Networks," *Proc. ACM SIGCOMM,* Aug. 2002.

[8] A. Crespo and H. Garcia-Molina, "Routing Indices for Peer-to-Peer Systems," *Proc. IEEE Int'l Conf. Distributed Computing Systems (ICDCS),* July 2002.

[9] A. Crespo and H. Garcia-Molina, "Semantic Overlay Networks for P2P Systems," technical report, Computer Science Dept., Stanford Univ., Oct. 2002.

[10] F. Dabek, M.F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Wide-Area Cooperative Storage with CFS," *Proc. ACM Symp. Operating Systems Principles (SOSP),* Oct. 2001.

[11] *FreePastry,* http://www.cs.rice.edu/CS/Systems/Pastry/FreePastry/, 2001.

[12] P. Ganesan, Q. Sun, and H. Garcia-Molina, "YAPPERS: A Peer-to-Peer Lookup Service over Arbitrary Topology," *Proc. IEEE INFOCOM,* Apr. 2003.

[13] L. Guo, S. Jiang, L. Xiao, and X. Zhang, "Exploiting Content Localities for Efficient Search in P2P Systems," *Proc. 18th Ann. Conf. Distributed Computing (DISC),* Oct. 2004.

[14] A. Iamnitchi, M. Ripeanu, and I. Foster, "Small-World File-Sharing Communities," *Proc. IEEE INFOCOM,* Mar. 2004.

[15] J. Kubiatowicz et al., "OceanStore: An Architecture for Global-Scale Persistent Storage," *Proc. ACM Int'l Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS),* Nov. 2000.

[16] J. Li, B.T. Loo, J. Hellerstein, F. Kaashoek, D.R. Karger, and R. Morris, "On the Feasibility of Peer-to-Peer Web Indexing and Search," *Proc. Second Int'l Workshop Peer-to-Peer Systems (IPTPS),* Feb. 2003.

[17] M. Li, W.-C. Lee, and A. Sivasubramaniam, "Semantic Small World: An Overlay Network for Peer-to-Peer Search," *Proc. 12th IEEE Int'l Conf. Network Protocols (ICNP),* Oct. 2004.

[18] B.T. Loo, R. Huebsch, I. Stoica, and J. Hellerstein, "The Case for a Hybrid P2P Search Infrastructure," *Proc. Int'l Workshop Peer-to-Peer Systems (IPTPS),* Feb. 2004.

[19] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and Replication in Unstructured Peer-to-Peer Networks," *Proc. ACM Int'l Conf. Supercomputing (ICS),* June 2002.

[20] R. Mahajan, M. Castro, and A. Rowstron, "Controlling the Cost of Reliability in Peer-to-Peer Overlays," *Proc. Int'l Workshop Peer-to-Peer Systems (IPTPS '03),* Feb. 2003.

[21] M. Castro, M. Costa, and A. Rowstron, "Debunking Some Myths about Structured and Unstructured Overlays," *Proc. USENIX Symp. Networked Systems Design and Implementation (NSDI),* May 2005.

[22] A. Muthitacharoen, R. Morris, T. Gil, and B. Chen, "Ivy: A Read/Write Peer-to-Peer File System," *Proc. USENIX Symp. Operating Systems Design and Implementation (OSDI),* Dec. 2002.

[23] S. Patro and Y.C. Hu, "Transparent Query Caching in Peer-to-Peer Overlay Networks," *Proc. 17th Int'l Parallel and Distributed Processing Symp. (IPDPS),* Apr. 2003.

[24] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A Scalable Content-Addressable Network," *Proc. ACM SIGCOMM,* Aug. 2001.

[25] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Application-Level Multicast Using Content-Addressable Networks," *Proc. Third Int'l Workshop Networked Group Comm. (NGC),* Nov. 2001.

[26] P. Reynolds and A. Vahdat, "Efficient Peer-to-Peer Keyword Searching," *Proc. ACM/IFIP/USENIX Middleware,* June 2003.

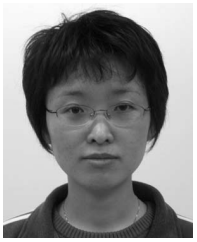
[27] S.C. Rhea and J. Kubiatowicz, "Probabilistic Routing and Location," *Proc. IEEE INFOCOM,* June 2002.

[28] M. Ripeanu, I. Foster, and A. Iamnitchi, "Mapping the Gnutella Network: Properties of Large-Scale Peer-to-Peer Systems and Implications for System," *IEEE Internet Computing J.,* vol. 6, no. 1, 2002.

[29] A. Rowstron and P. Druschel, "PAST: A Large-Scale, Persistent Peer-to-Peer Storage Utility," *Proc. ACM Symp. Operating Systems Principles (SOSP),* Oct. 2001.

[30] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," *Proc. ACM/IFIP/USENIX Middleware,* Nov. 2001.

[31] N. Sarshar, P.O. Boykin, and V.P. Roychowdhury, "Scalable Percolation Search in Power Law Networks," *Proc. IEEE Fourth Int'l Conf. Peer-to-Peer Computing (P2P),* Aug. 2004.
[32] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling Churn in a DHT," *Proc. USENIX Ann. Technical Conf. (ATC),* June 2004.
[33] K. Sripanidkulchai, B. Maggs, and H. Zhang, "Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems," *Proc. IEEE INFOCOM,* Apr. 2003.
[34] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," *Proc. ACM SIGCOMM,* Aug. 2001.
[35] Q. Sun and H. Garcia-Molina, "Partial Lookup Services," *Proc. 23rd IEEE Int'l Conf. Distributed Computing Systems (ICDCS),* May 2003.
[36] C. Tang and S. Dwarkadas, "Hybrid Global-Local Indexing for Efficient Peer-to-Peer Information Retrieval," *Proc. USENIX Symp. Networked Systems Design and Implementation (NSDI),* Mar. 2004.
[37] C. Tang, Z. Xu, and S. Dwarkadas, "Peer-to-Peer Information Retrieval Using Self-Organizing Semantic Overlay Networks," *Proc. ACM SIGCOMM,* Aug. 2003.
[38] C. Tang, Z. Xu, and M. Mahalingam, "pSearch: Information Retrieval in Structured Overlays," *Proc. ACM HotNets,* Oct. 2002.
[39] C. Wang, L. Xiao, Y. Liu, and P. Zheng, "Distributed Caching and Adaptive Search in Multilayer P2P Networks," *Proc. IEEE 24th Int'l Conf. Distributed Computing Systems (ICDCS),* Mar. 2004.
[40] B. Yang and H. Garcia-Molina, "Improving Search in Peer-to-Peer Systems," *Proc. IEEE 22nd Int'l Conf. Distributed Computing Systems (ICDCS),* July 2002.
[41] B. Yang and H. Garcia-Molina, "Designing a Super-Peer Network," *Proc. IEEE Int'l Conf. Data Eng. (ICDE),* Mar. 2003.
[42] R. Zhang and Y.C. Hu, "Borg: A Hybrid Protocol for Scalable Application-Level Multicast in Peer-to-Peer Systems," *Proc. ACM Int'l Workshop Network and Operating Systems Support for Digital Audio and Video (NOSSDAV),* June 2003.
[43] B.Y. Zhao, J.D. Kubiatowicz, and A.D. Joseph, "Tapestry: An Infrastructure for Fault-Resilient Wide-Area Location and Routing," Technical Report UCB//CSD-01-1141, Univ. of California Berkeley, Apr. 2001.
[44] Y. Zhu, H. Wang, and Y. Hu, "A Super-Peer Based Lookup in Highly Structured Peer-to-Peer Networks," *Proc. Int'l Conf. Parallel and Distributed Computing Systems (PDCS),* Aug. 2003.
[45] S.Q. Zhuang, B.Y. Zhao, A.D. Joseph, R.H. Katz, and J. Kubiatowicz, "Bayeux: An Architecture for Scalable and Fault-Tolerant Wide-Area Data Dissemination," *Proc. ACM Int'l Workshop Network and Operating Systems Support for Digital Audio and Video (NOSSDAV),* June 2001.



Rongmei Zhang received the BS degree in 1998 and the MS degree in 2000 from Tsinghua University, Beijing China, both in electrical engineering. She is currently a PhD candidate in computer engineering at Purdue University. Her research interests lie broadly in distributed systems and networking. In particular, she has worked on network measurement, Internet content distribution, multimedia systems and networking, peer-to-peer systems, and distributed computing. She is a member of the ACM and the IEEE.

Y. Charlie Hu (S '90, M '03, SM '07) received the MS and MPhil degrees from Yale University in 1992 and the PhD degree in computer science from Harvard University in 1997. He is an associate professor of electrical and computer engineering and computer science at Purdue University. From 1997 to 2001, he was a research scientist at Rice University. Dr. Hu's research interests include operating systems, distributed systems, networking, and parallel computing. He has published more than 100 papers in these areas. Dr. Hu received the Honda Initiation Grant Award in 2002 and the US National Science Foundation (NSF) CAREER Award in 2003. He served as a TPC vice chair for the International Conference on Parallel Processing in 2004 and IEEE ICDCS in 2007, and a cofounder and TPC cochair for the International Workshop on Mobile Peer-to-Peer Computing. Dr. Hu is a member of USENIX and the ACM, and a senior member of the IEEE.