Assisted Peer-to-Peer Search with Partial Indexing

Rongmei Zhang and Y. Charlie Hu
Purdue University
West Lafayette, IN 47907
{rongmei, ychu}@purdue.edu

***Abstract*— This paper proposes to improve search in unstructured peer-to-peer (P2P) overlay networks by building a partial index of shared data. 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 both the Web and P2P file sharing traces show that the assisted search with a lightweight partial indexing service can significantly improve the success rate and search speed in locating data, while inducing less traffic overhead than Gnutella and a hit-rate based protocol in unstructured P2P systems.**

Keywords: **Peer-to-Peer, search, indexing**

I. Introduction

In the past few years peer-to-peer (P2P) applications 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 involves flooding the P2P network. This decentralized approach is highly robust to failures or node transience. Since there is no explicit control over the network topology or data placement, these P2P networks are usually referred to as "unstructured". The performance of search in unstructured P2P overlays can be improved by using smarter search or data replication algorithms [5], [22], [26], [6]. Other recent work [4], [20], [27], [3] exploit the inherent locality in peer interests or the heterogeneity in peer capacity.

Meanwhile, a new class of P2P systems have been proposed [14], [21], [19], [29]. These P2P overlays effectively implement Distributed Hash Tables (DHTs), and search is performed by looking up the DHTs. In these "structured" P2P networks, data items can be located in a small number of hops without incurring excessive overhead. More recently, DHTs have been proposed to implement multiple-keyword search [15] or full-text search [24], [23].

In addition to the search algorithms, P2P file sharing systems also differ in whether they involve indexing of shared files. 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, local-scale indices are maintained by more resourceful peers on behalf of other nearby peers. In structured P2P systems, inverted lists of documents are published to the DHT under single terms [15], [23] or term (semantic) vectors [24]. Search is conducted by looking up the DHT-based index. In summary, so far indexing either is required for all data items or does not exist at all in P2P file sharing systems. Besides, if indexing is performed, it is only used for searching.

In this paper, we propose a DHT-based partial indexing scheme. Instead of indexing all data items owned by each peer, only a portion of them are registered with the index. The partial index has three complementary purposes. First, it helps peers to find shared interests. Peers are organized based on their interests in a similar way as proposed in [4], [20]. However, 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 similar interests. Peers that share common interests connect to each other as neighbors in the unstructured overlay. Because of the locality of interests, a query is more likely to be satisfied by nearby peers.

Second, the index also provides search hints for those data difficult to locate even after interest-based clustering. 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 (keywords), are also registered with the index. During search,

if the first attempt returns no hits 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 also benefit from more sophisticated search algorithms, e.g., random walks [13]. 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. Since the assisted search protocol is built on an unstructured overlay, it is also highly resilient to node failures or transient population changes. If one connection in the search overlay is lost, the index overlay is queried for candidates to replace the failed neighbor. Therefore, isolated failures are not likely to collapse the entire network, although affected peers may temporarily experience degraded performance.

The implementation of the indexing service can be separate from the unstructured search overlay. In this paper, the index is maintained by all peers of the unstructured overlay and it is implemented by a structured P2P overlay so that interest information or search hints can be retrieved quickly with small overhead. In other words, the search overlay and the index overlay are implemented on top of the same physical nodes, and peers can participate in both overlays.

The assisted search protocol as proposed above faces two key challenges. (i) "Interests" as presented to the index overlay should capture the major characteristics of the data that a peer shares with others, and unpopular data should be identified fairly accurately. Section III discusses how peer interests and data popularity can be determined locally by individual peers. (ii) The index information carried by the index overlay should be lightweight and the traffic for maintaining the index should be low.

The rest of the paper is organized as follows. Section II discusses related work. Section III presents the design of our proposed assisted search protocol. Section IV discusses the design of our trace-driven experiments and the evaluation results are presented in Section V. Section VI draws conclusions and discusses future work.

II. Related Work

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

A. 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 naive search algorithm based on flooding. In [13], an algorithm based on random walks is proposed to resolve queries. It can reduce the amount of network traffic by two orders of magnitude while the search speed is sacrificed slightly. Several strategies aimed at improving search efficiency are also studied in [26], including expanding ring search and a variation of random walks. The assisted search algorithm presented in this paper can also benefit from applying these mechanisms. For example, queries can be forwarded to a selected subset instead of all neighbors.

In [6], distributed routing indices are used to direct queries towards where they are more likely to be satisfied. Assisted search also utilizes indexing. However, the partial index is maintained by a logically separate overlay on behalf of the peers. The locality embedded in human interests has been recognized for its effectiveness in guiding search queries [4], [20]. Assisted search also leverages this locality but instead of gradually learning from history, peers express their interests explicitly and actively seek partners via the intermediate index overlay. More recently, the pattern and properties of file sharing between peers with common interests are also studied using "data sharing graphs" in [9].

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 [27], [3]. These efforts are orthogonal to our main ideas and we do not elaborate on issues regarding peer heterogeneity in this paper.

In [13], [5], data replication strategies in Gnutella-like P2P networks are discussed. By explicitly controlling the replication of data items based on their popularity, search scope can be reduced. Alternative data placement approaches and their impacts on search performance are also studied in [22]. In this paper, we focus on the search algorithm and assume the default replication strategy: the data is replicated by the querier upon a successful search.

B. Search in Structured and Hybrid P2P Overlays

While much of the effort has been directed towards better search algorithms in unstructured P2P overlays, various applications and services have been built based on structured P2P overlays, such as archival storage systems [7], [18], [10] and group communication mechanisms [2], [30], [28]. The feasibility of providing full-text Web search using P2P networks is studied in [11]. 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 towards 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 multi-keyword search in structured overlays [15]. pSearch [25], [24] implements full-text search based on CAN [14]. Both data and queries are represented by term (semantic) vectors and search is conducted as matching in a multi-dimensional Cartesian space. eSearch [23] proposes a hybrid global-local indexing scheme for full-text search; documents together with their full term lists are published only under the top keywords.

Probabilistic location [16] proposes a hybrid search scheme: it uses a lossy distributed index to locate data close to the querier and falls back on the default DHT-based algorithm if the probabilistic location fails. Probabilistic location enhances the default structured P2P routing, while our approach is opposite in that search is performed in the unstructured P2P overlay with the assistance of a structured index overlay.

Yappers [8] is also a hybrid approach to P2P search. Both data and peers are mapped to a small number of buckets in the logical space and each peer builds a small DHT of nearby neighbors based on this mapping. Both publishing and querying are guided by the DHTs so that only those peers that fall in the same bucket as the data are involved. This binning of data and peers is "blind" (by hashing the names) and does not consider data content.

Very recently, the work in [12] was brought to our attention. In [12], 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.

III. Design

A. Definitions

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

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 name "Smooth Operator", the singer "Sade", and the musical genre "jazz", etc. These meta-data are used in specifying queries. For example, if the peer receives a query featuring "Sade" in the search criteria, the song will be returned as a hit. 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. For example, if the peer has a large number of songs by "Sade" in its data repository, then its major interests can be partially characterized by "Sade". 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 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).

B. 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. 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 similar 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 [4], [20], this approach exploits the locality in human interests.

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 there may exist a small portion of queries that cannot 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 the 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. For example, while properties such as the singer "Sade" and the genre "jazz" are likely to become local interests, based on the amount of corresponding data, a property that is more specific to an individual data item such as the song title "Smooth Operator" cannot be selected as a representative interest for registration. This property may be determined as unpopular and registered to the index overlay if the

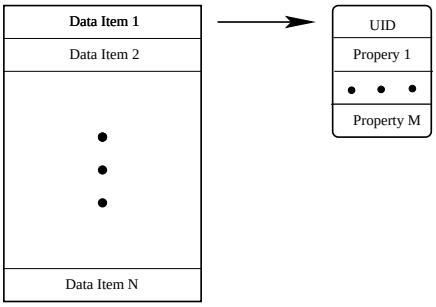

Fig. 1. Data possessed by a peer. Each data item contains one or more properties.

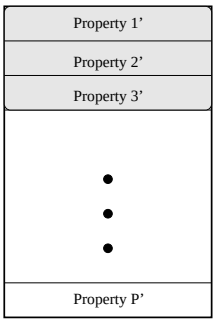

Fig. 2. Merged properties of local data items, sorted according to the frequency of appearance. Top ranked properties (highlighted) become the local interests of the peer.

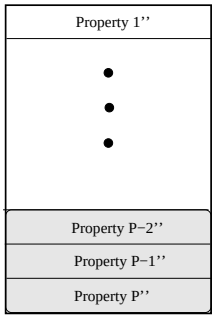

Fig. 3. Merged properties of local data items, sorted according to the frequency of appearance in received queries. Bottom ranked properties (highlighted) that are not part of local interests become the locally-observed unpopular properties.

"name" is hardly seen in any passing queries. 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.

C. Overlay 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 [19] 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 similar interests through a *lookup (key)* operation in the structured index overlay. For example, a peer can look up "jazz" by setting the key of the query message using the hash value of "jazz". This query message is received by the index overlay node that maintains pointers to those nodes that are also interested in "jazz". As the reply containing such nodes is received, the peer initiates connections to them, i.e., they become neighbors in the search overlay. Thus, the search overlay reflects the association of interests between connected peers. In other words, peers are clustered according to common interests.

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.

D. Overlay Maintenance

In order to exploit shared interests, peers first need to determine their local interests. As a peer accumulates its data repository by issuing queries and then downloading from other peers, its interests are automatically reflected by its data repository. The most prominent features can then be extracted from the properties of its data possession. This can be achieved by ranking the properties according to the number of data items that can be characterized by them and picking the top ranking properties as representing interests. In addition, peers with larger data repositories can register more interests, if there are more properties associated with large number of data items in their local data repository. 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 an *insert (key, value)* operation in the structured index overlay. The registries are soft-state and updated periodically because peer interests and their shared data may change over time. Lost registries due to node failures are automatically restored by subsequent registrations from the search overlay, i.e., registrations are directed to the next alive node that is responsible for the associated properties. The registries are also lightweight. Basically only the addresses of the registering peers need to be stored. It is optional for peers to provide extra information such as the local number of data items associated with the property. Besides, only the most representative interests, usually only a few, are registered for each peer.

Peers also continuously update their neighbor connections in the search overlay to reflect their changing interests. Periodically, a peer selects those properties of local data with the highest ranking and queries the index overlay for other peers sharing these interests. In this way the search overlay is able to adapt to changing peer interests. Usually more than one candidates for each interest property exist, and the peer can select the best one based on extra metrics. In this paper, the candidate that has the largest number of data items for the associated property is chosen.

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

E. Monitoring Popularity

The assisted search protocol also benefits from the registration of rare properties to the index overlay. Like the registration of interests, the decision to register unpopular properties is made by the owners 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 the search frequency of each property in the search criteria. The higher the search frequency, the more popular the property and the more likely that the associated data is well replicated. 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, the registries of unpopular properties are also soft-state and light-weight, i.e., the records can only contain pointers to the data providers. There is a trade-off between the number of the registries for unpopular properties maintained by the index overlay and the ability to search unpopular data. This trade-off can be controlled by the threshold of "unpopularity".

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", and relevant peers should be notified to stop registering the same property as unpopular in the future. This allows peers to improve the accuracy of their local popularity estimation and to avoid overloading the index overlay with unnecessary registrations.

F. 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 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. For example, both "jazz" and "1990s" are good candidates if the peer is seeking jazz music from the 1990s. The destinations returned by the index overlay include both those peers that have strong interests in the corresponding properties and those registrations based on data popularity estimation. 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 against their data repositories, 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 desired 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 speed and the top candidates are contacted first.

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

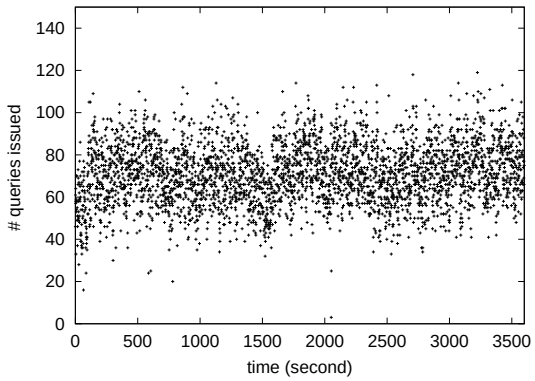

Fig. 4. Query rate: number of queries per second (Web trace).

IV. Experimental Methodology

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

A. Query Workload

We use two different traces from real file-sharing applications. The first trace captures Web requests from a large corporate network and the second is collected from real-world Gnutella peers.

1) Web Trace: we use the Web trace obtained from the proxy logs at Boeing [1]. The original trace consists of the Web requests recorded by one proxy over the period of one day (24 hours) on March 1st, 1999. The query load varies with time and we extracted a one-hour segment from the peak period during the middle of the day (see Fig. 4).

The peers in the Web trace are treated as peers in the simulations. We model queries as Web requests, data items as the URLs in the requests, and properties as the corresponding hostnames. Since both URLs and hostnames are anonymized, we can not model partial match. Each URL corresponds to only one hostname, so we can not model multiple properties per data item. However, the URL-hostname relationship captures the association between data items sharing the same property: they are more likely to be accessed by the same peers. The original traces were parsed to remove those URLs only accessed by one peer since they are irrelevant to the sharing of content in P2P systems. After parsing, the trace has 5879 peers, 29724 URLs, and 1970 hostnames.

Fig. 5 shows the cumulative distribution of hostnames regarding the number of URLs that share the same hostname. More than 50% of hostnames correspond to at most 3 URLs

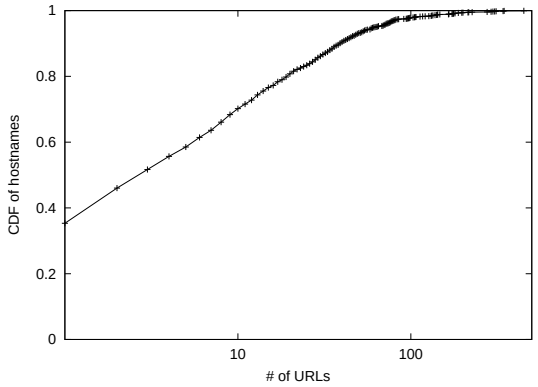

Fig. 5. CDF of hostnames vs number of URLs belonging to the hostname (Web trace).

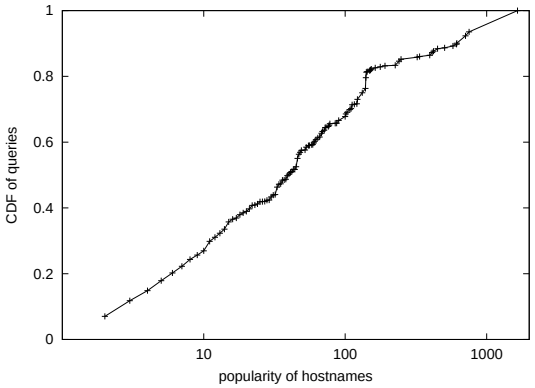

Fig. 7. CDF of queries vs popularity of hostnames (number of peers accessing the hostname) (Web trace).

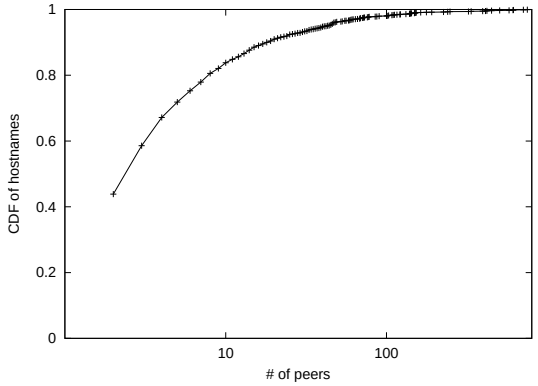

Fig. 6. CDF of hostnames vs number of peers accessing the hostname (Web trace).

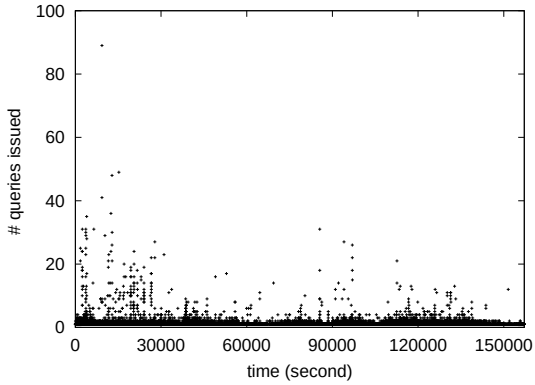

Fig. 8. Query rate: number of queries per second (Gnutella trace).

each. This implies that for a large amount of properties (hostnames), each represents only a few data items (URLs) and therefore these properties are unlikely to be selected as peer interests.

The trace is also highly skewed in terms of data popularity, as shown in Fig. 6. For example, about 43% hostnames are requested by only 2 peers each. A significant amount of queries are made for these extremely unpopular hostnames (Fig. 7): about 20% of all requests are made for those hostnames only accessed by 4 or less peers. The large volume of unpopular data in the trace imposes great challenges to the search protocol.

Before running the simulations, the first copy of a data item is placed at the peer that makes the first request for it. Replication at the querier follows a successful query for the data item, i.e., the actual URL in the trace is downloaded. We also assume that each copy (including the original data item) is available for sharing with other peers from the time it is downloaded.

2) Gnutella Trace: The Gnutella trace was collected using modified Gtk-Gnutella version 0.93, which is a Unix Gnutella peer software based on the GIMP Toolkit (GTK+). Changes were made to dump Query and QueryHit messages, but the behavior of the peer was not affected in any way. Four peers 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 (June 25–26, 2004). In this period all passing Query and QueryHit messages were dumped.

The collected data was then processed to extract all queries issued by immediate neighbors of the collecting peers, plus all corresponding replies. The Gnutella protocol allows the trace collector to identify those queries originated from immediate neighbors from either the TTL field or the Hops field in the Gnutella packet header. Since many peers use non-default TTL value, we used the Hops field. Fig. 8 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 and partial matching. The processed trace consists of 1540 peers, 8019 files and 26959 words (keywords). Fig. 9 shows that the majority (over 80%) of queries and filenames contains 10 keywords or less.

Fig. 10 shows the frequency of keywords appearing in filenames in the trace: more than half keywords are seen in only one filename and more than 80% in at most 5 filenames. In this regard, the Gnutella trace is similar to the Web trace: a large number of keywords are not likely to be selected as peer interests.

Fig. 11 and Fig. 12 depict the popularity of keywords in the Gnutella trace. First, almost 60% of keywords are queried by only one peer and more than 75% by at most two (Fig. 11). Second, the appearance frequency of keywords in query mes-

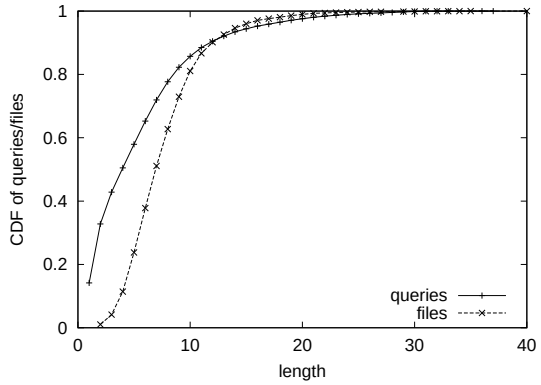

Fig. 9. CDF of query/file length (number of keywords) (Gnutella trace).

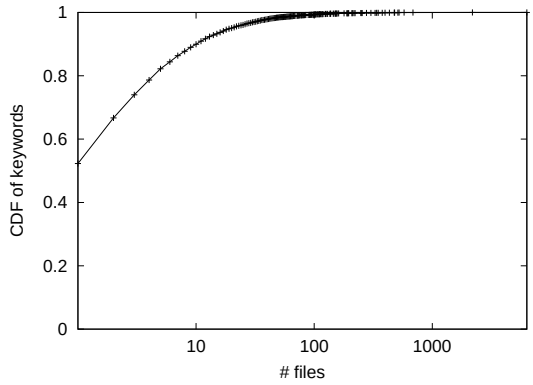

Fig. 10. CDF of keywords vs number of filenames containing the keyword (Gnutella trace).

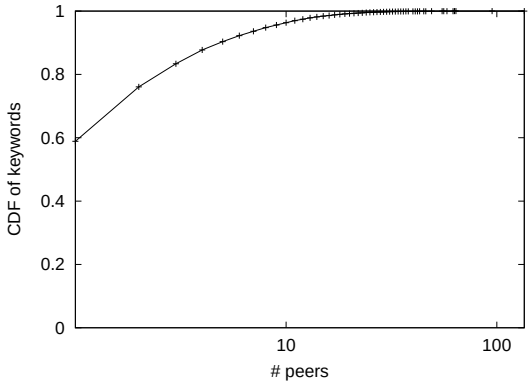

Fig. 11. CDF of keywords vs number of peers querying the keyword (Gnutella trace).

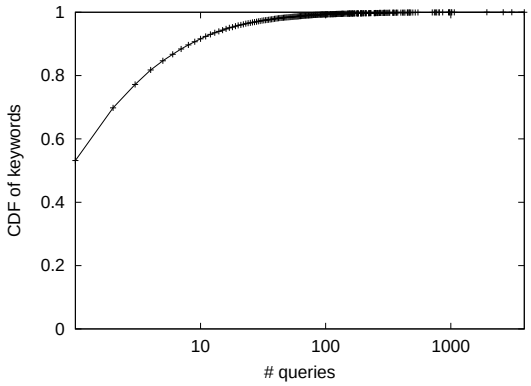

Fig. 12. CDF of keywords vs number of queries containing the keyword (Gnutella trace).

sages follows a similar distribution (Fig. 12). These statistics suggest that the popularity of keywords is highly skewed in the Gnutella trace, similar to the Web trace. In fact, the popularity skew in the Gnutella trace is more significant for individual keywords than in the Web trace. For example, in the Web trace, about 71% of hostnames are accessed by 5 or fewer peers, as opposed to as many as 90% of keywords in the Gnutella trace. Therefore, there exist many more rarely seen data properties in the Gnutella trace than in the Web trace.

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. Similarly to the Web trace, we assume that each file is available for sharing from the moment of being replicated.

B. Protocol Configurations

In our experiments, we assume that the index overlay and the search overlay completely overlap, i.e., each peer participates in both overlays. The index overlay is a Pastry network and peers join it with random node identifiers.

The assisted search protocol are 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 peer interests (labeled "interest-based" in Fig.13-26). The second configuration also organizes the search overlay based on peer interests. In addition, the querier is given a second chance if the first search attempt fails to satisfy a query, and the index overlay is contacted for search hints, i.e., the potential destinations for the second try (labeled "interest-based x 2"). The third configuration adds to the second configuration the registrations of unpopular data properties (labeled "interest+popularity-based x 2"). Both pointers from interest-based registrations and popularity-induced registrations are returned as equally possible destinations for the second search attempts.

In the search overlay, we assume uniform connectivity for all peers. Each peer is connected to 4 neighbors, which is close to the average degree of a Gnutella network [17]. 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 III-D. Depending on the configuration, peers can also periodically update their local interests and their selection of unpopular data with the index overlay. For the Web trace, the interval of updating is set to 1.0 minute for peer connections and 0.5 minute for local interests and unpopular data. For the Gnutella trace, the intervals are 5.0 minutes and 1.0 minute respectively, because this trace runs much longer than the Web trace.

The search overlay implements the Gnutella protocol, e.g., a query is forwarded to all neighbors except the one from which the query comes from. The flooding of queries is controlled by a TTL. In particular, the TTL is set to 6 for the Web trace and 4 for the Gnutella trace since the second trace has fewer nodes than the first. A query is considered successful if at least

one reply is received, either from the first or the second search attempt.

In the simulation, a property that has more than the local average number of data items per property is registered with the index overlay. A property is considered unpopular if the number of associated queries is less than the local average number of recorded queries per property. In addition, the property corresponds to at least one data item in the local data repository and does not overlap with registered local interests. The index overlay also provides feedbacks to the search overlay regarding data unpopularity observations. Specifically, for the Gnutella trace, if an indexing node sees 10 or more registrations for a particular property, it labels this property not qualified as unpopular. The threshold is set to 50 for the Web trace.

We compare the 3 versions of the assisted search protocol with the Gnutella protocol (labeled "random" in Fig. 13-25). We also compare the "interest-based" version with a "history-based" counterpart. The "history-based" scheme only involves the search overlay and 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. This "history-based" protocol updates neighbors in a way similar to how the shortcuts are selected in [20]. The interval of updating neighbor connections is set to be the same as in the "interest-based" scheme.

C. Performance Metrics

We use the following metrics to evaluate the performance of different search protocols:

a) Success Rate: The success rate measures the effectiveness of the search protocol and is defined as the percentage of queries that are resolved successfully.

b) Search Delay: The search delay is measured by the time elapsed until the first reply is received at the querier and is only defined for successfully resolved queries.

c) Search Scope: The search scope is defined by the fraction of peers contacted in resolving a query. It characterizes the messaging overhead in the search overlay.

d) Overhead: The overhead for the search overlay is defined as the average percentage of local data properties that have to be submitted at each peer during the routine of updating interests and unpopular data to the index overlay. The overhead for the index overlay, on the other hand, is measured by the average number of registries maintained at each node.

e) Interest Overlap: We also measure the interest locality in the search overlay using the number of overlapping data properties between neighboring peers. Interest locality reflects the effectiveness of a search protocol in aggregating peers with similar data sharing interests.

V. Experiment Results

This section presents the simulation results of evaluating the performance of the assisted search protocol. The simulation was run using both the Web and the Gnutella traces.

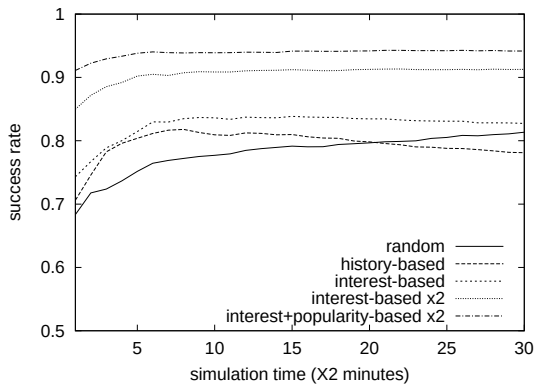

Fig. 13. Success rate (Web trace).

A. Success Rate

Fig. 13 shows the success rate of queries from the Web trace. First, the "interest-based" scheme is more effective than the "history-based" scheme in locating data items. Towards the end of the simulation, the "history-based" scheme even deteriorates and the success rate of search falls below that of the random protocol (the reason is explained in the next paragraph). On the other hand, the "interest-based" scheme always achieves higher success rate (about 5-20%) then the random protocol. Second, more than half of the queries that cannot be resolved by flooding the interest-based overlay can be satisfied by consulting the index overlay for search hints. Third, providing search hints for unpopular data via the index overlay can further reduce search failures to around 5%.

Fig. 14 shows the corresponding search scope for the Web traces. While search scope remains roughly constant for the random search protocol, it declines over time for both "history-based" and "interest-based" schemes and the decline rate is higher for the "history-based" scheme. This explains why its success rate falls even below that of the random protocol (Fig. 13): Intuitively, the opportunity of finding the data being searched for is higher as the query reaches a larger population. The shrinking search scope is a result of applying the interest-based or history-based clustering. The "random" configuration generates even distribution of connections among all peers. The "interest-based" and "history-based" schemes cause a small number of peers to have more (incoming) connections than others; we will quantify this side effect later in this section.

The performance gain by applying the "interest-base" scheme can be further explained by Fig. 15, which characterizes the degree of interest overlap between peers and their direct neighbors in the search overlay. By using the "interest-based" or "history-based" clustering, the amount of shared data properties increases substantially between neighbors during the simulation, compared with the default protocol. By explicitly extracting and exchanging peer interests, the "interest-based" scheme is even more effective in inducing interest locality than the "history-based" scheme.

Fig. 16 shows the success rate of search using the Gnutella trace. We introduce the "optimal" success rate, the success rate if queries are flooded to the entire network. The "optimal" success rate is 100% for the Web trace, since there exists a copy for each data item (URL) being queried in the trace after ini-

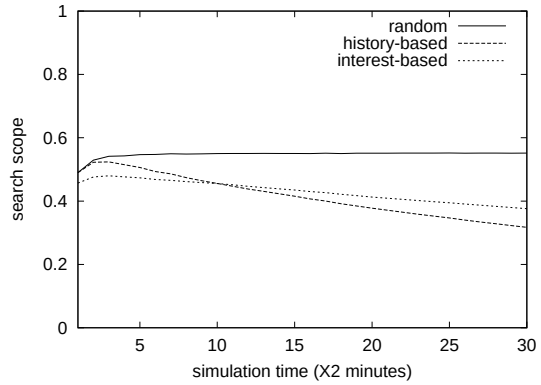

Fig. 14. Search scope (Web trace).

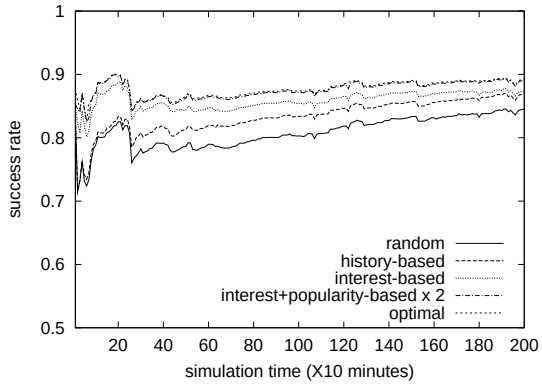

Fig. 16. Success rate (Gnutella trace).

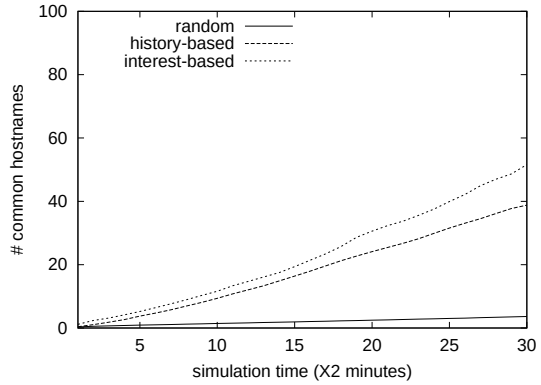

Fig. 15. Common data properties (hostnames) with immediate neighbors (Web trace).

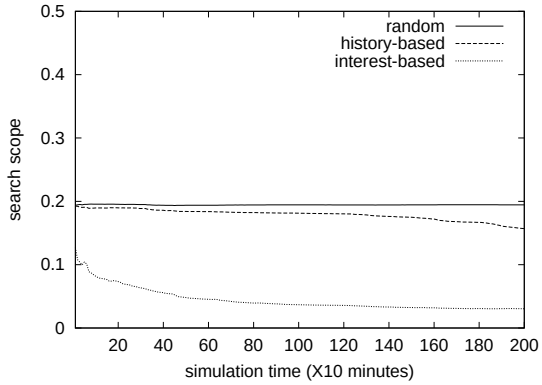

Fig. 17. Search scope (Gnutella trace).

tialization. However, the maximum achievable success rate is about 90% for the Gnutella trace, as shown in Fig. 16. This is because that there exist no matching files in the Gnutella trace for a small number (about 10%) of queries.

The random search protocol can achieve about 80% success rate with about 20% search scope (Fig. 17). The "interest-based" protocol can achieve close to optimal results (about 8% higher than the random protocol) with as low as 3% search scope. The "history-based" scheme can also improve the search success rate by about 5%, compared to the random protocol. However, the "interest-based" scheme is notably more effective in exploring interest locality: it achieves higher success rate with much smaller search scope.

It is also important to point out that after applying the "interest-based" scheme, it helps little to give a second chance to those queries that cannot be fulfilled by the first search attempt. This suggests that interest clustering in the search overlay is highly effective in discovering shared interests and it can achieve the best possible results by exploiting interest locality. Therefore for clarity, we omit the results for "interest-based x2" here, since they mostly overlap with those of the "interest-based" option. However, the search performance can be further improved by second chances if unpopular data are also registered to the index overlay. Fig. 16 shows that it can achieve the optimal success rate, as the results of "interest+popularity-based x 2" overlap with the "optimal" results.

Fig. 18 describes the evolving interest locality between immediate neighbors throughout the simulation. Compared with Fig. 15, the sheer number of shared data properties is higher, due to the fact of multiple keywords per data item. Secondly, the gap between different configurations is more significant than in the Web trace simulation. Fig. 17 and Fig 18 together show that the assisted search protocol is effective in discovering and exploring interest locality, and it is potentially more powerful for P2P file-sharing applications represented by the Gnutella trace.

B. Search Delay

The search delay for both traces is depicted in Fig. 19 and Fig. 20. First, the search delay improves over time, as peers receive the first reply for a query more and more quickly. Second, both the "interest-based" and the "history-based" protocols reduce the search delay, compared with the original Gnutella protocol; the "interest-based" scheme achieves the lowest delay among all schemes simulated. This further demonstrates the benefits of peer clustering based on their explicit interests.

While the success rate is improved with second search attempts, the average search delay also becomes longer. This is because those queries satisfied by second tries experience longer delay, which includes the time out (2xTTL) for the first try, the delay to retrieve search hints from the index overlay, and the delay to finally resolve the query through the second try. However, for both traces, the average delay with the second search is still lower than that of the default Gnutella because a large portion of queries can be resolved quickly through the first search attempts.

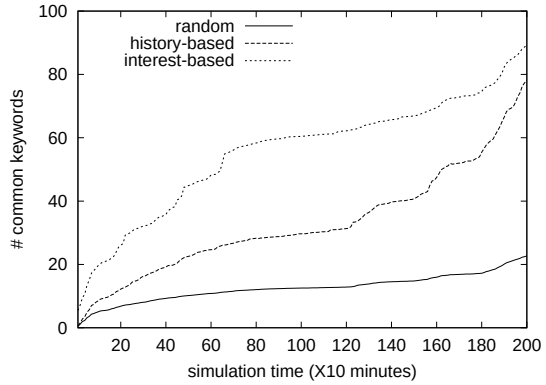

Fig. 18. Common data properties (keywords) with immediate neighbors (Gnutella trace).

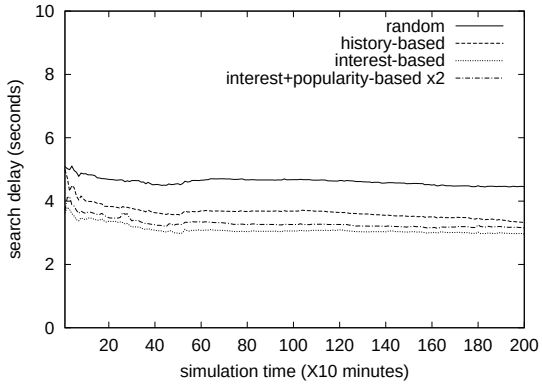

Fig. 20. Search delay (Gnutella trace).

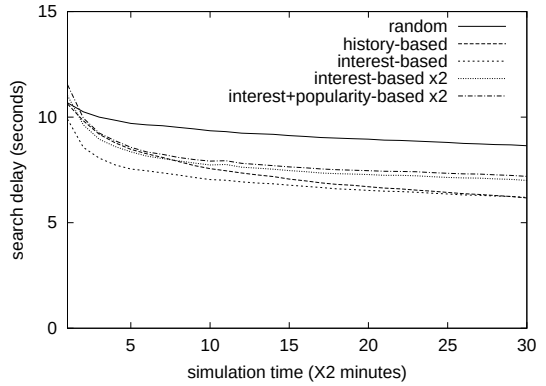

Fig. 19. Search delay (Web trace).

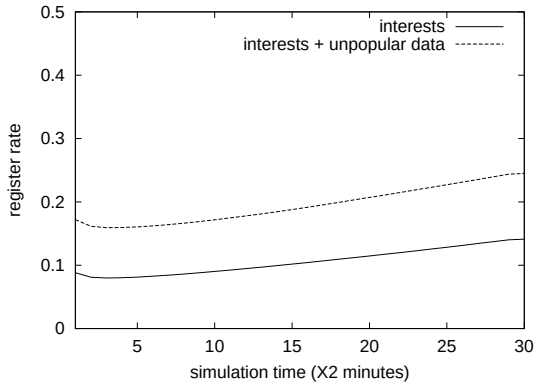

Fig. 21. Overhead in the search overlay: average registration rate (Web trace).

C. Overhead

The overhead of the assisted search protocol is characterized by the percentage of data properties registered (i.e., registration rate) to the index overlay by the search overlay (Fig. 21 and Fig. 22) and the number of registries maintained by the index overlay (Fig. 23 and Fig. 24).

For the Web trace, the registration rate stays below 15% for peer interests. It is increased to below 25% when unpopular data are also registered. For the P2P trace, the registration rate is much lower: it tops at about 6.6% when both favorite and unpopular data are counted. However, in this case unpopular data becomes the more significant contributor to the overall registration rate. These differences between the two sets of results are consistent with the analysis of the properties of the traces (Section IV).

Fig. 23 and Fig. 24 show that the number of registries maintained by the index overlay is higher for the Gnutella trace, since each data item corresponds to multiple properties (as many as 40). The majority of registries come from unpopular data from the search overlay, and this can also be explained by the large number of unpopular keywords from the Gnutella trace (Fig. 11 and Fig. 12). In addition, due to the feedback mechanism, the number of unpopular data registries even declines slightly over time, as local estimations of peers are refined based on global observations of the index overlay.

D. Impact on the Overlay Topology

In our experiments, every peer in the search overlay is connected to 4 other peers, i.e., the outdegrees of peers are uniform. However, their indegrees can vary. The difference can be amplified when applying the “interest-based” or “history-based” clustering, as those with larger data repositories are likely to become connected to more peers. Fig. 25 shows the CDF of peer indegree for both schemes, compared with the random search protocol for the Gnutella trace. The Web trace yields similar results. While all nodes have indegrees of at most 11 for the random protocol, the results from both the “interest-based” and “history-based” schemes have long thin tails. This difference in the characteristics of the overlay topology is due to applying interest-based or history-based clustering and explains the reduced search scope for these two schemes, as shown previously.

E. Summary

In our experiments, we have used two traces that are different in two major aspects. First, queries in the Web trace come from one enterprise network, while the Gnutella trace is not confined within any organization. Therefore, the Web trace exhibits stronger interest locality between all participating peers, as we have profiled previously. Second, interests are reflected by accessing the same Web site for the Web trace. In this regard, the Gnutella trace resembles text-based search more closely, where interests are determined by repeated keywords.

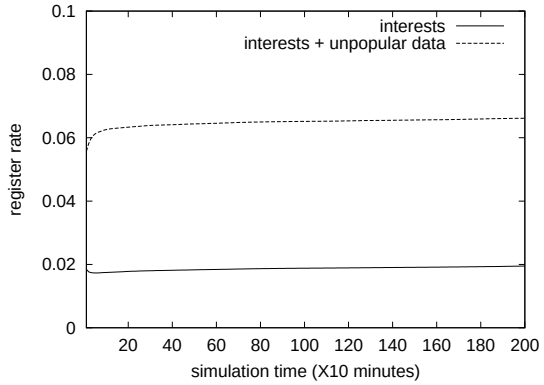

Fig. 22. Overhead in the search overlay: average registration rate (Gnutella trace).

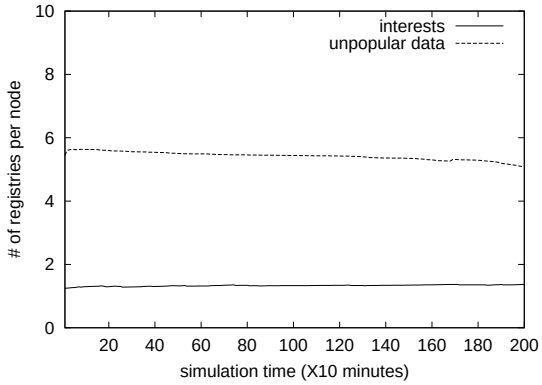

Fig. 24. Overhead in the index overlay: average registries per node (Gnutella trace).

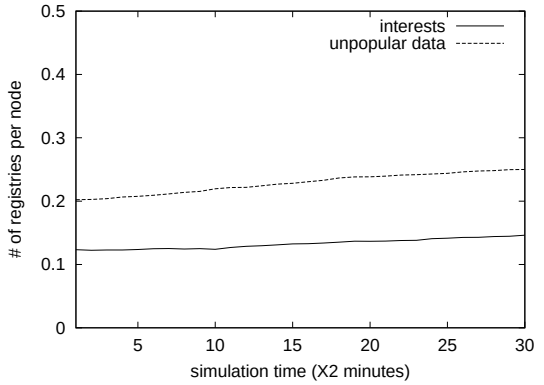

Fig. 23. Overhead in the index overlay: average registries per node (Web trace).

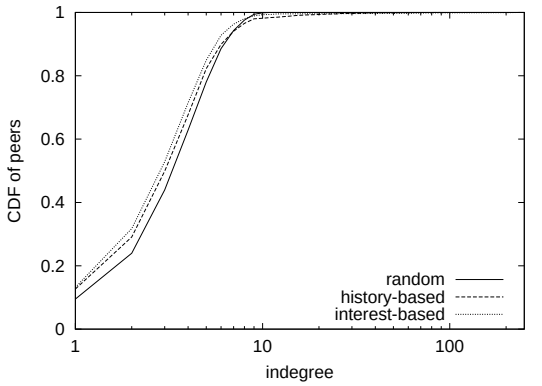

Fig. 25. Node indegree (Gnutella trace).

Running the assisted search protocol on top of both traces have demonstrated the effectiveness of the protocol in diverse content sharing contexts. The interests as extracted from local data repositories indeed reflect the querying and data sharing characteristics of peers. Even for the Gnutella trace where shared interests between peers are more scarce, the proposed mechanism can capture and exploit existing interest locality to achieve close to optimal search performance. Moreover, simple techniques based on local observations can effectively identify unpopular data.

In addition to the success rate, the assisted search protocol is also beneficial in improving the search speed and in controlling the messaging overhead during search.

VI. Conclusions and Future Work

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. It exploits the locality between individual peer interests and meanwhile recognizes global data popularity. The assisted search protocol achieves higher search efficiency and scalability 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 traces from both the Web and 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.

Our proposed assisted search protocol exploits the long-term interests of peers by taking into account all their data possession, regardless of the freshness, i.e., when the data items were actually acquired. Nevertheless, the protocol can be easily modified to account for the temporal locality in peer interests, i.e., by exploiting the short-term interest locality. In our future work, we plan to investigate the impact of short-term interests on the performance of the assisted search protocol with partial indexing.

One variation for the assisted search protocol is to first estimate the possibility of success using the first search attempt (i.e., TTL controlled flooding). If the query is unlikely to be resolved by the first try, the index overlay can be consulted directly. For instance, if none of the properties being searched for appears in a significant number of data items in the local data repository, e.g., fewer than the local average number of data items per property, the index overlay should be contacted for search hints first. If no search hints are returned, or the search hints can not satisfy the query, the peer can also be given a second chance by resorting to the default TTL controlled flooding. Therefore, this alternative protocol can achieve the same success rate in locating data. However, it is unclear how the search latency and overhead are affected. We plan to study the alternative search protocol as part of our future work.

ACKNOWLEDGMENT

We thank David Yau and the anonymous reviewers for their helpful comments. This work was supported by an NSF CAREER award ACI-0238379.

REFERENCES

[1] Web Traces of Logs, 1999. http://www.web-caching.com/traces-logs.html.

[2] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron. Scribe: A Large-Scale and Decentralized Application-Level Multicast Infrastructure. *IEEE JSAC on Communication*, October 2002.

[3] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making Gnutella-like P2P Systems Scalable. In *Proceedings of ACM SIGCOMM*, August 2003.

[4] E. Cohen, A. Fiat, and H. Kaplan. Associative Search in Peer-to-Peer Networks: Harnessing Latent Semantics. In *Proceedings of IEEE INFOCOM*, April 2003.

[5] E. Cohen and S. Shenker. Replication Strategies in Unstructured Peer-to-Peer Networks. In *Proceedings of ACM SIGCOMM*, August 2002.

[6] A. Crespo and H. Garcia-Molina. Routing Indices for Peer-to-Peer Systems. In *Proceedings of IEEE ICDCS*, July 2002.

[7] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-Area Cooperative Storage with CFS. In *Proceedings of ACM SOSP*, October 2001.

[8] P. Ganesan, Q. Sun, and H. Garcia-Molina. YAPPERS: A Peer-to-Peer Lookup Service over Arbitrary Topology. In *Proceedings of IEEE INFOCOM*, April 2003.

[9] A. Iamnitchi, M. Ripeanu, and I. Foster. Small-World File-Sharing Communities. In *Proceedings of IEEE INFOCOM*, March 2004.

[10] J. Kubiatowicz et al. OceanStore: An Architecture for Global-Scale Persistent Storage. In *Proceedings of ACM ASPLOS*, November 2000.

[11] 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. In *Proceeding of IPTPS*, February 2003.

[12] B. T. Loo, R. Huebsch, I. Stoica, and J. Hellerstein. The Case for a Hybrid P2P Search Infrastructure. In *Proceedings of IPTPS*, February 2004.

[13] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and Replication in Unstructured Peer-to-Peer Networks. In *Proceedings of ACM ICS*, June 2002.

[14] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A Scalable Content-Addressable Network. In *Proceedings of ACM SIGCOMM*, August 2001.

[15] P. Reynolds and A. Vahdat. Efficient Peer-to-Peer Keyword Searching. In *Proceedings of ACM/IFIP/USENIX Middleware*, June 2003.

[16] S. C. Rhea and J. Kubiatowicz. Probabilistic Routing and Location. In *Proceedings of IEEE INFOCOM*, June 2002.

[17] 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 Journal*, 6(1), 2002.

[18] A. Rowstron and P. Druschel. PAST: A Large-Scale, Persistent Peer-to-Peer Storage Utility. In *Proceedings of ACM SOSP*, October 2001.

[19] A. Rowstron and P. Druschel. Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-peer Systems. In *Proceedings of ACM/IFIP/USENIX Middleware*, November 2001.

[20] K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems. In *Proceedings of IEEE INFOCOM*, April 2003.

[21] 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 ACM SIGCOMM*, August 2001.

[22] Q. Sun and H. Garcia-Molina. Partial Lookup Services. In *Proceedings of IEEE ICDCS*, May 2003.

[23] C. Tang and S. Dwarkadas. Hybrid Global-Local Indexing for Efficient Peer-to-Peer Information Retrieval. In *Proceedings of USENIX NSDI*, March 2004.

[24] C. Tang, Z. Xu, and S. Dwarkadas. Peer-to-Peer Information Retrieval Using Self-Organizing Semantic Overlay Networks. In *Proceedings of ACM SIGCOMM*, August 2003.

[25] C. Tang, Z. Xu, and M. Mahalingam. pSearch: Information Retrieval in Structured Overlays. In *Proceedings of ACM HotNets*, October 2002.

[26] B. Yang and H. Garcia-Molina. Improving Search in Peer-to-Peer Systems. In *Proceedings of IEEE ICDCS*, July 2002.

[27] B. Yang and H. Garcia-Molina. Designing a Super-Peer Network. In *Proceedings of IEEE ICDE*, March 2003.

[28] R. Zhang and Y. C. Hu. Borg: A Hybrid Protocol for Scalable Application-Level Multicast in Peer-to-Peer Systems. In *Proceedings of ACM NOSSDAV*, June 2003.

[29] 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, U. C. Berkeley, April 2001.

[30] 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. In *Proceedings of ACM NOSSDAV*, June 2001.