Neighborhood Signatures for Searching P2P Networks

Mei Li Wang-Chien Lee Anand Sivasubramaniam
Department of Computer Science and Engineering
Pennsylvania State University
University Park, PA 16802
E-Mail: {meli, wlee, anand}@cse.psu.edu

Abstract

Overlay networks have received a lot of attention due to the recent wide-spread use of peer-to-peer (P2P) applications such as SETI, Napster, Gnutella, and Morpheus. Through replications at numerous peers, digital content can be distributed or exchanged with high resilience and availability. However, existing P2P applications incur excessive overhead on network traffic. For example, Gnutella, which broadcasts queries to search shared content, suffers from an overwhelming volume of query and reply messages. In this paper, we investigate the issues of trading-off storage space at peers to reduce network overhead. We propose signatures for focussing searches along selected network paths, and introduce three schemes - CN, PN-S *and* PN-A, *to facilitate efficient searching of shared content in P2P networks. With little storage overhead, these signatures improve the performance of content search and thus significantly reduce the volume of network traffic. Extensive simulations based on uniform and power-law network topologies are conducted to evaluate the performance of our proposal with existing P2P content search methods, including Gnutella, Random Walk, and Local Index. Results show that PN-A gives much better performance at a small storage cost.*

Key words: Peer-to-peer networks, signature files, Internet content search, information sharing, indexing techniques

1 Introduction

The advent of facilities such as Napster [5] and Gnutella [3] has made the Internet a popular medium for the widespread exchange of resources and voluminous information between thousands of users. In contrast to traditional client-server computing models, where one or more nodes are specifically designated as service providers, these Peer-to-Peer (P2P) systems as they are called, can employ the host nodes to themselves acting as servers for other nodes. Despite avoiding centralized server bottlenecks and single points of failure, these decentralized systems present interesting challenges when one needs to locate data items present in one or more of these numerous host nodes. Search techniques need to be efficient in locating items without requiring too many node traversals and without flooding the network, be able to tolerate nodes failing or dropping out, and be scalable to thousands of nodes. The centralized server in Napster [5], which maintains a global index for all data items in the network, defeats the fundamental rationale of a P2P system.

One could control the placement of data on the nodes, index the placed data (in a distributed manner), and/or exploit the topology of the network to perform some kind of search ordering which can help in getting to the requested data items. CAN [18], Chord [21], Pastry [19], Tapestry [24] and P-Grid [7, 8] are examples of systems using such a strategy to control the number of hops that need to be traversed to get to the data items without flooding the network. However, the completely decentralized nature of P2P systems, which allow nodes and data items to come and go at will, makes such techniques less suitable for these kinds of unstructured and dynamic environments, and are consequently not under consideration here.

There are two main solution strategies that have been proposed/explored for searching in these kinds of decentralized environments without relying on the network topology and data placement:

- **Strategy 1:** The first strategy is to let messages poll nodes, without having any idea of the data held by the destination node, till the required items are found. Gnutella [3] uses such a strategy, and broadcasts search messages to a set

of nodes, which may in turn broadcast to other nodes if they are not able to satisfy the request. While this may seem appealing for unstructured systems, allowing full freedom for nodes to appear and disappear without requiring additional overheads, the downside is the network overload due to the flooding of search messages [20]. A study by Clip2 shows that nodes quickly become overloaded, with query traffic increasing linearly with the size of the network, and this strategy is simply not suitable for P2P networks where hosts may only have a 56K modem connection [1]. One way of alleviating network load is to simply have one or a small number of outstanding messages per request, which traverses one node after another (usually randomly since we are not relying on any structured ordering) till the data items are found. Such a mechanism, termed a *random walk* [9, 17], reduces the flooding problem but can result in large latencies (hops) to satisfy a request, or not be able to answer queries within bounded times.

- **Strategy 2:** This strategy maintains additional information in the network nodes (which Strategy 1 does not require) in order to reduce network traffic and/or the number of hop visits. Consequently, messages are directed specifically along paths that are expected to be more productive. The information is typically some kind of index over the data that is contained by either nearby neighbors [12, 16, 23] or within hierarchical clusters [4]. Morpheus [4] uses a clustering approach, with supernodes selected for each cluster that maintain an index of data held by the nodes within that cluster. However, broadcasting to other supernodes is employed when the required data is not present within the local cluster. Indexing on the data held by neighbor nodes [12, 16, 23] has been explored in depth. This requires determining what indexes (keys) to construct a priori, consulting neighbors when they join/leave, and constraining the attributes for the search based on the predetermined indexes, in addition to the high space cost that is incurred in storing the index itself.

While Strategy 2 seems attractive in terms of message traffic, the downside is that the additional storage required can weigh on the actual implementation. In fact, one could argue that with infinite capacity, it is possible to replicate all availability information at every node, potentially leading to very efficient searches. While this is one extreme, at the other end are schemes in Strategy 1, which do not require any storage but incur high network costs for searches. Schemes which index neighborhood data in Strategy 2 fall in-between, trading off the space requirements for the network overheads. We believe that this trade-off offers a rich space of mechanisms to explore, and we present/demonstrate a novel approach that uses *signature files* which can provide better message traffic behavior at a lower storage cost than index-based mechanisms within this space.

Signature files have been extensively studied in information retrieval [14]. Attributes of a data record are hashed to binary strings and the resulting strings of all attributes are superimposed. Signatures from different records can further be superimposed for a convenient representation of several data items as a signature file. A query searching for particular data items can also be similarly composed as a signature. This mechanism can thus be an effective way of representing several data items in a much smaller space (than indexing on them). A node can maintain signature files for data in neighboring nodes than maintaining indexes for them. However, while indexing can exactly tell whether a data item is present or not, signature files may give false indications at times, i.e. the signatures can match but the data item may not actually be present - denoted as *false positive*. This can result in a message getting forwarded more than it should in certain cases, and the space devoted to the signature can influence the probability of false positives, which in turn affects performance. Signatures have the nice property that it does not give false negatives, i.e. signature does not match but data is present.

This paper presents three novel ways of using signatures, namely *CN*, *PN-S* and *PN-A*, to represent neighborhood data at network nodes for optimizing searches in P2P systems. Their merits in reducing network traffic are extensively evaluated for search queries, node join and leave operations, with different criteria used to control the search latencies (minimum number of retrieved data items, maximum number of search depth) denoted as *stop conditions*. These schemes are compared with the current state-of-the-art in the two strategies - Gnutella [3] and Random Walk [9, 17] for strategy 1, and Local Index [23] for strategy 2 - to illustrate their pros and cons. When storage of remote node information is not an option, one has to resort to schemes such as random walk (or perhaps even broadcasts as in Gnutella). As can be expected at the other end, when we have the luxury of no storage limitations at nodes, indexing could be the better option since it does not have any false positives. Together with validating such intuitive observations, our simulations show that the signature approaches are much better than these two alternatives for most reasonable storage space availability assumptions on host nodes. They perform better - requiring much lower network bandwidth - for both stop conditions.

There have been techniques suggested to store additional information on intermediate nodes - caching query results [2, 20] or maintaining history about prior operations [23] (number of returned query results or prior response times) - to reduce network traffic. While the effectiveness of such enhancements is quite dependent on query patterns and their locality, they are orthogonal to this work and can be used in conjunction with our signature schemes to further control network traffic. In addition, there are services aiming at indexing and ranking all the content available in a P2P network [13]. This service

uses a technique called *bloom filter*, which is similar to the signature technique used in this paper, to efficiently summary the indexed terms. However, the focus is on providing a content search and retrieval engine rather than reducing the network traffic.

The rest of the paper is organized as follows. Next section presents the P2P system model and metrics. In Section 3, we presents details on our signature based search mechanisms, together with a qualitative comparison of the other two prior approaches. Section 4 gives the experimental setup for the evaluation and Section 5 details results from experiments with two different search criteria. Finally, Section 6 summarizes the contributions of this paper and outlines directions for future work.

2 Peer-to-Peer System Framework

A Peer-to-Peer (P2P) network[1] consists of numerous nodes (called *peers*) which connect to each other directly or indirectly. The peers provide information resources to be shared with other peers. The shared information could be digital files such as music clips, images, pdf documents, or other forms of digital content. The P2P network is established by logical connections among the participating peers. Because the peers are distributed over the Internet, they communicate by sending messages through these logical connections of the overlay network on top of the Internet. Since the whole network is built through connections among the peers, its topology may change dynamically due to constant joins and leaves of the peers, namely *peer join* and *peer leave*, in the network. In addition, the shared information changes dynamically since the peers may update the digital content they offer, named as *peer update*.

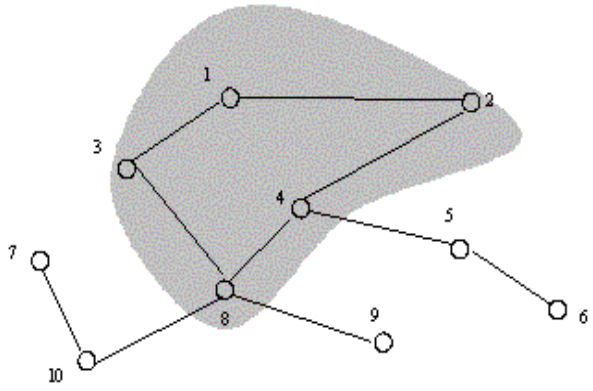


Figure 1. A partial snapshot of a P2P network.

Figure 1 shows a partial snapshot of a P2P network. In this figure, we use a vertex to represent a node (i.e., a peer) of the overlay P2P network and an edge to denote the connection between two peers. When a peer, A, has a direct connection with another peer, B, we call these two peers *neighbors*. In the network, a peer may reach another peer via one or more sequences of connections, called *paths*. The *path length* can be obtained by counting hops of connections. The *distance* between two peers is the minimal path length between them. For example, as illustrated in Figure 1, there are two paths of length 3 and 4, respectively, between node 1 and node 9. Thus, the distance between node 1 and node 9 is 3.

Traditionally, a peer knows its neighbors through direct connections. In this paper, we generalize the concept of neighbors to a *neighborhood*, which includes all the peers reachable within a given distance from the peer. Following this definition, the *neighborhood radius* refers to the distance from a peer to the edge of its neighborhood. The shaded area shown in Figure 1 illustrates a neighborhood of radius 2 (consists of node 2, 3, 4, 8) for node 1.

[1] We use the terms, P2P systems, P2P networks and P2P applications, where appropriate. However, they are mostly interchangeable in the context of this paper.

2.1 Searches in P2P Networks

As mentioned above, P2P networks have been widely used for digital content sharing among the participating peers. Thus, efficient search of the shared content is one of the primary functions of the P2P networks. A user may initiate a search of digital content from any peer in the network. When a peer receives a *search* message with some criteria specified by keywords, key attributes and/or metadata, it first performs the search over the local content. If the search can be satisfied locally, a pointer to the result is returned to the user. Meanwhile, the search message is forwarded to all or a subset of its neighbors to extend the search. In order to prevent indefinite search in the P2P network, a stop condition is usually specified in the query. The following are two conditions that typically used to stop excessive spreading of a search:

- **Maximum search depth**: This stop condition is used in Gnutella [3]. A preset time-to-live value (TTL) is included in the search message to keep track of maximum search depth remaining. Each time a search message is forwarded to a neighbor, its TTL is decreased by 1. Once the TTL reaches 0, a search message is dropped.
- **Minimum number of results**: This stop condition is used in random walk [9, 17]. Different from the first case, the total number of results (TNR) found so far is included in the search message. Each time a result is found, the TNR is increased by 1. The search is stopped when the total number of results reaches a system defined value.

2.2 Existing Search Techniques

Various philosophies have been applied to content search in P2P networks. As was explained earlier, they can be broadly classified into two strategies. In the following, we briefly go over two specific mechanisms for strategy 1, and one mechanism for strategy 2, which are used in comparison with our proposal.

2.2.1 Strategy 1: Search with no additional storage requirements

Here, the nodes simply send out search messages without any idea of whether the destination has the required data (since they do not maintain any auxiliary information for their neighbors). The following two techniques capture two representative implementation of this strategy:

- **Flooding**

  The simplest search technique, adopted by Gnutella, is to flood the P2P network with search messages. In Gnutella, a search message is forwarded by a peer to all its neighbors until the message reaches a certain preset distance (i.e., maximum search depth is used as the stop condition). In other words, a search initiated by a peer in the Gnutella network will be extended to a wide area of the network (with a system preset radius) from the peer. This technique is simple and efficient for small scale networks. However, excessive traffic can be caused by this flooding approach when the network is large, which may seriously deteriorate the performance of the underlying Internet.

- **Random walk**

  To address the issue of excessive traffic caused by the flooding search, random walk chooses to forward a search message from a peer to one or more of its neighbors, which are chosen randomly. This search typically uses the minimum number of results as the stop condition. Our examination in this paper uses a single path random walker which goes through one neighbor at a time. A primary advantage of this approach is that the search traffic may be significantly reduced by performing single line search over a sequence of neighbors instead of performing parallel search over a wide area of network, as is done by Gnutella. This approach also allows the search to go beyond a pre-defined neighborhood. The random walk has been evaluated by Ademic et al. [9] and Lv et al. [17], who show that this consumes much less bandwidth than flooding.

2.2.2 Strategy 2: Index-based Search

Whether it be flooding or random walk, it would definitely make sense to NOT forward search messages to a neighbor that does not have a satisfying search result. This requires that the sender maintains some additional information about its neighbors, which is kept by mechanisms in strategy 2.

In the traditional computing environment, indices have been used extensively for disk-based content search [10, 11]. Indices are efficient for searching the well-organized information stored in disks or main memory because they provide a

guidance regarding the whereabouts of the data being searched. Since the indices are usually constructed in accordance with the organization and storage of data, they can direct the search deterministically.

While the digital content available in a P2P network is not offered in an organized fashion, the ideas behind indices can be used to construct mechanisms that facilitate directed search (in contrast to blind search in flooding and random walk approaches) in a relatively decentralized manner (since we are only concerned with nearby neighbors). To address the excessive traffic issue caused by Gnutella, several index structures have been proposed for searching P2P networks [12, 16, 23]. The basic idea is that, by maintaining auxiliary information over digital content offered within a neighborhood, a peer can perform the search on behalf of all peers located within its neighborhood. *Local Index* is an index structure that indexes the neighborhood of a peer [23]. Each peer in the P2P network maintains an index over its neighborhood. In this way, searches can be limited to a subset of the search neighborhood and the traffic is reduced.

The problem with the index approach is that it is only suitable for searches that query against the index key(s) used to construct the index. It is inefficient for keyword search, which is used for most Internet search, over content with various searchable attributes. In addition, local index incurs high storage overhead when the size of the data set increases, which is a trend since more and more information is made available for sharing and exchange on P2P networks.

2.3 Metrics

While the P2P networks enable establishment of cyber communities that diversify the use of Internet, the technology actually causes communication deterioration. Since the primary issue investigated in this paper is the trading-off of storage space for reducing network traffic, we use the following metrics to evaluate various P2P search techniques discussed in this paper:

- **Total message volume:** For signature schema (as well as index schema), in addition to the traffic incurred for search, additional traffic is incurred as construction/maintenance cost of auxiliary information at peers. In order to have fair comparison among different approaches, we use *total message volume*, the product of message number times the size of different messages (including search messages and messages incurred for signature maintenance during peer join/leave/update), as one of the performance metrics.
- **Storage size:** The other metric we use in this paper is the size of storage used for maintaining auxiliary information (signature/index) at peer nodes averaged over the total number of peers.

3. Neighborhood Signatures

Signature methods have been widely used in information retrieval. A signature of a digital document, called *data signature*, is basically a bit vector generated by first hashing the attribute values and/or content of the document into bit strings and then superimposing them together[2].

To facilitate search, a *search signature* is generated in a similar way as a data signature based on the search criteria (e.g., keywords) specified by a user. This search signature is matched against data signatures by performing a bitwise *AND* operation. When the result is not the same as the search signature, the corresponding document can be ignored. Otherwise, there are two possible cases. First for every bit set in the search signature, the corresponding bit in the data signature is also set, and the document is indeed what the search is looking for. This case is called a *true match*. In the second case, even though the bits may match, the document itself does not match the search criteria. This case, which occurs due to certain combinations of bit strings generated from various attribute values, keywords, or document content, is called *false positive*. Obviously the matched documents still need to be checked against the search criteria to distinguish a true match from a false positive.

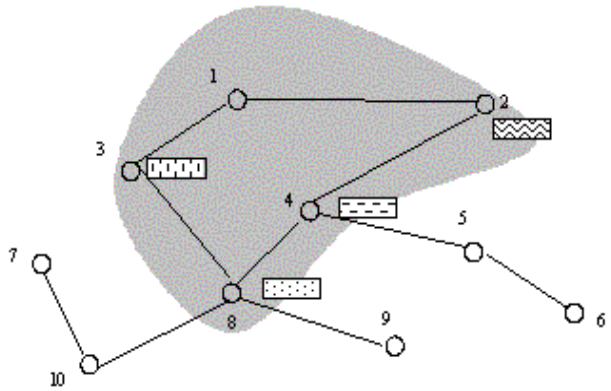
In this section, we propose extending signature methods for searching in P2P networks, and propose three neighborhood signature schemes, namely, *PN-S*, *PN-A*, and *CN*, to index the content offered within the neighborhood of a peer. This helps direct the search to a subset of the nodes, while not requiring as much storage as index approaches. We first describe the formation of the signatures for each scheme and then provide detailed algorithms for search and signature maintenance under various scenarios (i.e., peer join, peer leave, and peer update). Finally, we compare our proposals with existing techniques qualitatively.

[2]In this paper, we use the term, *superimposing* of two signatures, to denote a bitwise *OR* of the two bitstrings of the individual signatures.

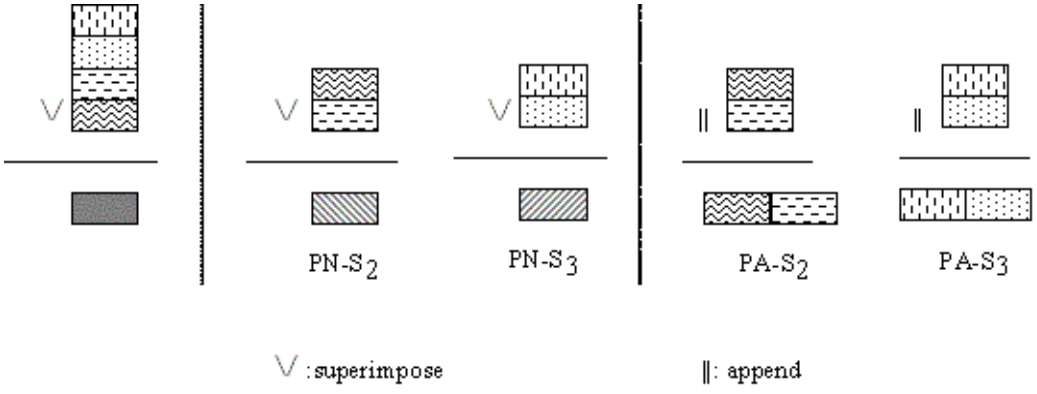
3.1. Signature Schemes

Before proceeding to introduce the proposed signature schemes, we first assume that a *local signature* is created at each peer of a P2P network to index the local content available in the peer. By doing this, search over the local content of a peer is processed efficiently. Furthermore, A peer may collect and maintain auxiliary information regarding to digital content available within a specific network distance (i.e., its neighborhood). Therefore, a peer will filter unsatisfiable search requests before forwarding them to a neighbor. Based on this idea, we propose three signature schemes classified as follows:

- **Complete Neighborhood (CN):** One intuitive approach is to index all the content available within the neighborhood of a peer. Thus, a *complete neighborhood (CN) signature* is generated by superimposing all the local signatures located within the neighborhood of a peer. Figure 2(a) shows the same partial snapshot of P2P network as Figure 1 with the local signatures of peer 2, 3, 4, and 8 represented by rectangle with different filling patterns. Figure 2(b) shows an example of a complete neighborhood signature for peer 1, which indexes all the content available at peers 2, 3, 4, and 8. By holding a complete neighborhood signature, a peer can determine whether the search should be extended in its neighborhood or simply forwarded to some peers outside of its neighborhood.



(a) Peers with local signatures



(b) CN (c) PN-S (d) PN-A

Figure 2. Illustration of neighborhood signature generation: (a) Peers with local signatures; (b) CN; (c) PN-S; (d) PN-A

.

- **Partial Neighborhood (PN):** While the CN scheme has the advantage of jumping out of a neighborhood when the search and neighborhood signatures do not match, it has to forward the search to all of its neighbors when there is a match between the neighborhood signature and search signature. Thus, instead of indexing the complete neighborhood, a signature can be generated to index a *partial neighborhood* branching from one of the neighbors directly connected to

a peer. A *partial neighborhood signature* is generated for each of the neighbors. The goal is to increase the precision of search within the neighborhood of a peer. The search will only be extended to the neighbors whose associated partial neighborhood signatures have a match with the search signature. There are two alternatives for generating partial neighborhood signatures:

- **Superimpose (PN-S):** In this approach, we use the traditional superimposing technique. Thus, all the local signatures located within a neighborhood branch are compressed into one signature, called *PN-S signature*. Figure 2(c) shows that peer 1 has 2 PN-S signatures, while PN-S_2 indexes all the contents available at peer 2 and 4 and PN-S_3 indexes the contents available at peer 3 and 8.
- **Append (PN-A):** The superimposing technique has been shown to be effective in compressing a large amount of index information while supporting efficient information filtering function. However, this compression comes at the cost of losing some information, i.e. when the PN-S signature at a node matches, it does not give an idea of which of the neighbors should be visited, resulting in searching all of peers that contributed to this superimposed signature. An alternative that we propose, called *PN-A Signature*, is to append (concatenate) all the local signatures within a branch of the neighborhood into a partial neighborhood signature [3]. When a search signature is matched with some sub-signatures within a PN-A signature, the search will only be forwarded to these peers associated with the matched sub-signatures. Figure 2(d) shows that peer 1 has 2 PN-A signatures, while PN-A_2 indexes all the contents available at peer 2 and 4 and PN-A_3 indexes the contents available at peer 3 and 8.

One could wonder why we bother considering PN-S, after having described the benefits of PN-A. The reason is that the former can take much less space, and allows us to find out which is a better alternative for a given space overhead at each node: (a) appending the individual signatures as in PN-A to fill up this space with each of the individual signatures being small (and not allowing better filtering), or (b) allowing a much larger signature for each peer and the information loss coming only from the superimposition. These trade-offs will be evaluated in later experiments.

3.2. Search Algorithms and Stop Conditions

The neighborhood signature schemes are generic mechanisms that can adapt to different search philosophies and protocols. As explained in Section 2, the Gnutella flooding approach uses the maximum search depth while the random walk uses the minimum number of results as the criteria for limiting message propagation. In order to compare the signature schemes with local index in Strategy 2, and Gnutella (which uses flooding) as well as random walk (which randomly chooses nodes to visit one after another) in Strategy 1, we discuss the signature based search algorithms for the following two philosophies: flooding/maximum-depth and single-path/minimum-result. For clarity of our presentation, we use x to denote the radius of a neighborhood.

3.2.1 Flooding Search

In this section, we describe how a peer utilizes neighborhood signatures to perform searches based on flooding search. Since the search algorithms for the three proposed signature schemes are similar, we use Algorithm 1 to detail the flooding search at a peer based on CN signatures and point out the differences for the PN signatures afterwards.

This algorithm is invoked when a search message is initiated or received at a peer node of a P2P network. This search message comes with a time-to-live (*TTL*) counter which was preset to the maximum search depth that this message may be forwarded. The peer first computes a search signature to compare with the local signature. If there is a match, the content at this peer node is examined to determine whether this is a a true match or a false positive. If this is a true match, a pointer to the result is returned back to the peer from which this node got the request. Next, the peer checks the *TTL* to see whether the edge of the search neighborhood has been reached (i.e. $TTL = 0$), and if so, the search message is dropped. Otherwise, the search signature is compared with the neighborhood signature. If there is a match, the search is extended to all the neighbors by forwarding the message with *TTL* decreased by 1. If the search signature does not match with the neighborhood signature, the peers located within x hops away (the neighborhood) need not be checked. As a result, the search message is dropped when $TTL \leq x$. The search should be processed only at the peers $x + 1$ hops away. Here we assume that a peer has the knowledge of its peers at $x + 1$ hops away so that it may forward the search messages directly.

[3] An append-based CN signature can be generated by simply appending all the partial neighborhood signatures, and is thus not proposed as a separate method.

Algorithm 1 Flooding search based on complete neighborhood signatures.

Incoming Message: Search_Msg(*TTL*)
Local Variables: Local_Sig, Neighborhood_Sig, Search_Sig
System Parameters: x {the neighborhood radius}
Procedure:

```
 1: compute Search_Sig based on Search_Msg.
 2: {check local content}
 3: if match(Search_Sig, Local_Sig) then
 4:    examine local content to verify whether this is a true match or not.
 5:    if true match then
 6:       return a pointer to the result back to the peer from which this node got the request.
 7:    end if
 8: end if
 9: {check whether reach an edge of the search area}
10: if TTL = 0 then
11:    stop
12: end if
13: {continue to search the neighborhood}
14: if match(Search_Sig, Neighborhood_Sig) then
15:    forward the message Search_Msg(TTL −1) to all the neighbors.
16: else
17:    if TTL > x then
18:       forward the message Search_Msg(TTL −x − 1) to all the neighbors located x + 1 hops away.
19:    end if
20: end if
```

The flooding search algorithms for the two partial neighborhood (PN) signature schemes are only slightly different from the one discussed above (refer to line 14-15). Due to space constraints, they are not listed separately in the paper. Instead of maintaining only one neighborhood signature for the whole neighborhood, a partial neighborhood signature is generated for each of the neighbors in these two schemes. The PN-S scheme compresses all the local signatures in a neighborhood branch into one signature (by superimposing), while the PN-A scheme enlists (appends) all the local signatures in a neighborhood branch into one signature. When a search signature matches with a PN-S signature, the search message is forwarded to the associated neighbor. Otherwise, the message is forward to the peers $x + 1$ hops away, located right outside of the partial neighborhood corresponding to the compared neighborhood signature.

The comparison of a search signature with a PN-A signature is performed by matching all the included local signatures. For every matched local signature, a search message is directly forwarded to the corresponding peer node. If the search signature does not match with a PN-A signature, the search message is forward to the peers $x + 1$ hops away, located right outside of the partial neighborhood corresponding to this neighborhood signature.

3.2.2 Single-Path Search

In this section, we describe how a peer utilizes neighborhood signatures to perform single-path search based on the minimum number of results as the stop condition (for comparison with random walk). Similar to flooding, we use Algorithm 2 to detail the single-path search at a peer based on CN signatures and point out the differences for the PN signatures afterwards. The main difference between single-path search and flooding search is that if all of the neighborhood signatures do not match with the search signature, a peer located $x+1$ hops away is randomly selected to extend the search. A system parameter *TNR* indicating total number of results found so far has the similar role as *TTL* in flooding search.

For CN signature, if the neighborhood signature matches with the search signature, all neighbors are possible candidates for true matches. In order to determine whether the match is a true match or not and how many results are there in the neighborhood, the search should be extended in the neighborhood for checking (called *neighborhood checking*). Since the neighborhood checking process is bound within the neighborhood instead of jumping out of the neighborhood, we use a different message, called as *Neighborhood_Check_Msg*, to selectively flood the neighborhood in order to expedite the checking process. Algorithm 3 details the processing of the neighborhood checking messages.

The difference among the single path search algorithms for CN, PN-S and PN-A is similar to what we observed for flooding search. If there are signature matches, the neighborhood checking messages are only forwarded to the neighbors with matching neighborhood signatures in PN-S, or directly to the peers with matching local signatures in PN-A.

Algorithm 2 Single-path search based on complete neighborhood signatures

Incoming Message: Search_Msg(*TNR*)
Local Variables: Local_Sig, Neighborhood_Sig, Search_Sig
System Parameters: x {the neighborhood radius}, R {the minimum number of result}, T {timeout}
Procedure:

```
1: compute Search_Sig based on Search_Msg.
2: {check local content}
3: if match(Search_Sig, Local_Sig) then
4:    examine local content to verify whether this is a true match or not.
5:    if true match then
6:       increase TNR by the number of results satisfied at this peer.
7:       return a pointer to the result back to the peer from which this node got the search message.
8:    end if
9: end if
10: {check whether found enough results}
11: if TNR ≥ R then
12:    stop
13: end if
14: {continue to search the neighborhood}
15: if match(Search_Sig, Neighborhood_Sig) then
16:    forward a neighborhood-checking message Neighborhood_Check_Msg(TNR, x) to all neighbors.
17:    wait for a T period of time to receive and forward result pointers back to the peer from which this node got the search message.
18:    increase TNR by the number of received result pointers.
19: end if
20: if TNR ≤ R then
21:    forward Search_Msg(TNR) to a randomly selected peer located x + 1 hops away.
22: end if
```

3.3 Signature Construction and Maintenance

After describing how the search is performed with neighborhood signatures, we next move on to discuss the construction and maintenance of these signatures. Basically, neighborhood signature(s) are constructed at a peer node when the peer newly joins a network. The neighborhood signatures of a peer will need re-constructions or updates when some peers join/leave its neighborhood or when some peers in its neighborhood (including itself) update their content. Thus, we describe the actions to be taken at peer join, peer leave, and peer update. Due to limited space, we only describe the general rules instead of giving detailed algorithms for each signature scheme.

- **Peer join:** A new peer informs its arrival by sending a join message including its local signature to the peers in the neighborhood. When a node receives such a join message, it first adds (either superimposes or appends) the local signature in the join message to the corresponding neighborhood signature, then sends back its own local signature to the new peer so that the new peer can construct neighborhood signatures. Besides this, some peers that were not in the neighborhood earlier, may be brought into this neighborhood through the connections of the newly joined node (when the new node joins the network through multiple connections and the neighborhood radius is greater than one). In this case, these peers also need to exchange signatures via the newly joined node to maintain the accuracy of their neighborhood signatures.

- **Peer leave:** When a peer leaves the network, it informs the neighbors by sending out a leave message. For PN-A, the leave message contains the node identifier of the leaving peer. The update on neighborhood signatures for PN-A only requires removing the signature of the leaving peer from the neighborhood signatures. For CN or PN-S, this step is more complicated. Since there is no simple way to remove the local signature of the leaving peer from the CN and PN-S signatures which are generated by superimposing, the affected peers in the neighborhood have to re-construct their neighborhood signatures from scratch. In order to construct a new CN neighborhood signature, the affected peer asks for individual signatures from the peers in the neighborhood by sending to these peers a pseudo join message. The pseudo join message functions similar to a real join message in that the receivers of both messages send back their local signatures, but differs in that when a peer receives a pseudo join message, it does not need to update its own signature. Slightly different from CN, for PN-S, the affected peers only need to ask for individual local signatures from the peers of the affected branch. For instance, assuming that the neighborhood radius is 2, when peer 6 in Figure 1 leaves the system, peers 4 and 5 are affected. In CN, peer 4 asks for local signatures from peers 1, 2, 3, 5, 8, 9 and 10, while peer

Algorithm 3 Neighborhood checking process based on complete neighborhood signatures

Incoming Message: Neighborhood_Check_Msg(*TNR, Hop_Count*)
Local Variables: Local_Sig, Neighborhood_Sig
System Parameters: x {the neighborhood radius}, R {the minimum number of result}
Procedure:

```
1: compute search_Sig based on Neighborhood_Check_Msg.
2: {check local content}
3: if match(Search_Sig, Local_Sig) then
4:    examine local content to verify whether this true match or not.
5:    if true match then
6:       increase TNR by the number of results satisfied at this peer.
7:       return a pointer to the result back to the peer from which this node got this neighborhood checking message.
8:    end if
9: end if
10: {check whether find enough results or reach the neighborhood edge}
11: if TNR ≥ R or Hop_Count = 0 then
12:    stop
13: end if
14: {continue to search the neighborhood}
15: if match(Search_Sig, Neighborhood_Sig) then
16:    forward Neighborhood_Check_Msg(TNR, Hop_Count − 1) to all neighbors.
17:    receive and forward result pointers back to the peer from which this node got the neighborhood checking message.
18: end if
```

5 asks for local signatures from peer 2, 4 and 8. In PN-S, for peer 4, only the neighborhood signature associated with peer 5 is affected; therefore, peer 4 only asks for local signatures from peer 5. For peer 5, the update of neighborhood signature only involves removing the neighborhood signature associated with peer 6.

- **Peer update:** When a peer updates its data content, the local signature is updated accordingly. The procedure for updating the neighborhood signatures for CN and PN-S is the same as peer leave since new neighborhood signatures need to be constructed again. For PN-A, the difference between the updated signature and the old signature are recorded in a *change* record, which is included in the update message, so that the affected peers can update their neighborhood signatures accordingly.

What we described above (and adopt in this paper) are called *eager update*. A peer involving join, leave and update proactively informs other peers located within its neighborhood to update the affected neighborhood signatures immediately. An alternative approach, called *lazy update*, is to postpone the updates on its neighborhood signatures till there is some other (routine) message exchange - such as ping-pong messages of Gnutella or even search messages. For instance, during these routine message exchanges, a peer can detect that a neighbor peer left the network through lack of reply from this node within a certain time period. Although it is possible that the neighbor peer is still in the network and cannot reply within a reasonable time due to severe network congestion or errors, we can still assume that the neighbor peer left the network and update the neighborhood signature in order to benefit future searches. There are many variations of such lazy update mechanisms which can be applied to peer-to-peer systems to improve the performance. Although lazy updates can reduce maintenance costs, there are many issues involved, such as data consistency maintenance, the ability to tolerate inconsistency, the optimal update delay etc. We intend to investigate such issues in future work, and in this paper we use the eager update method.

3.4. Qualitative comparison of different search schemes

Table 1 compares the signature approaches with Gnutella, random walk, and local index. Index and signature approaches can have low message volume since the search is only forwarded to a subset of peers in a neighborhood being searched. If the neighborhood radius is high, the message volume of index and signature can be reduced substantially. Random walk can involve many more messages when the data replication ratio in the network is relatively low. In addition to search message volume, both index and signature approaches have index/signature construction/maintenance overhead, which the other two do not have. This overhead is proportional to the index/signature size. Since the index size is much larger than signature size, the overhead for index is usually larger than for signatures.

Both index and signatures improve performance at the expense of extra storage. However, the storage requirement of signature is much lower than index. Moreover, the signature schemes can adapt to the available storage. In addition to lower

Table 1. Comparison between Gnutella flooding, Random Walk, index and signature approaches.

Methods	Total message volume		Limited to certain key attributes	Storage requirement
	Search	Join/Leave/Update		
Gnutella	high	none	no	none
Random walk	moderate	none	no	none
Index	low	moderate	yes	high
Signature	low	low-moderate	no	low

total message volume at a much smaller storage requirement, the signature approach is suitable for arbitrary attribute based search, keyword based text search, and content based search. In this aspect, indexing approach is not a good choice since it only supports some selected key attributes.

The downside of index and signatures are in the additional message exchanges for join/leave/update, and the effect of these can be offset when search is much more frequent. The trade-offs between these is investigated quantitatively in the rest of this paper.

4. Simulation Parameters

Simulation based experiments have been conducted to evaluate the performance of our proposed approaches with existing P2P search techniques such as Gnutella, random walk, and local index, using the flooding and single-path methods, which employ the maximum-depth and minimum-results stop conditions respectively, as described in Section 2.

We consider two different network topologies, *uniform* and *power-law*, which have been studied in related work as well [17, 23]. Based on [15], we set the power-law topology using an exponential efficiency of 1.4. In addition, by assuming that there are 1000 peers pre-existing in a P2P network, the simulations are initialized by generating signatures for these initial 1000 peers. Then, we inject a large number of operations - a randomized mix of search, peer join, peer leave, and peer update - into the P2P network in each experiment. The relative proportion of these operations has also been considered.

We vary several system parameters, such as neighborhood radius, storage size, key attribute size, number of data items at a peer, relative proportion of different operations, and the average number of replicas per data in the network. In subsequent discussions, *search/update ratio* indicates the proportion of search to the other operations requiring signature maintenance (i.e., peer join, peer leave and peer update), and *replication ratio* is defined as the number of replicas per data divided by the number of peers in the network.

Table 2 lists the parameters and their values used in the simulations and the justification for these choices is as follows:

- **System parameter settings:** The average number of shared files per peer has been observed to be around 340 in [23], and so we set number of data items per peer to 400. *Key attributes*, which contain the key value(s) of data items in various forms, e.g., binary music clip, keywords, integers, etc, are used for evaluation of search criteria. We use *size of key attribute* as an important parameter to characterize data items, since the size of data items is itself not a significant factor in differentiating the schemes under investigation, In most of our experiments, we use 4 bytes as a default for the size of key attribute (i.e., we assume a single value attribute unless specified). We also ran experiments by increasing the size of key attribute (i.e., to represent a multi-key composite attribute or a complex attribute with binary data such as music clip) and the number of data items per peer in order to observe their impacts on different search approaches. The average number of neighbors is set to 4, which is consistent with the average node degree in Gnutella [22]. In P2P network, if a message traveling through an edge reaches a peer that has seen the same message before, this edge closes a cycle and we call this kind of edges *redundancy edges*. On the average about 30% of the searches are dropped due to cycles in the network, so we set the ratio of redundant edges to 30%.

- **Stop condition settings:** The maximum search depth is set to 7 in Gnutella. However, some detailed studies on Gnutella networks concluded that the network diameter is about 4 [15]. So we set the maximum search depth to 4 for power-law networks. We ran some preliminary experiments and found out that in order to achieve the same coverage for searching in both network topologies, the maximum search depth should be set to 5 in uniform network. While the replication ratios are varied in the single-path search, we set the minimum number of results to 1 in these experiments.

Table 2. Parameter setting used in the simulations.

System parameters	
Number of data items per node	400
Size of the key attribute in a data item	4 bytes
Number of nodes in the network	1000
Number of neighbors per node	4
Ratio of redundant edges in the networks	30%
Stop conditions	
Maximum search depth	uniform topology: 5; power-law: 4
Minimum number of results	1
Message parameters	
Size of message head	80 bytes
Size of join message	80 + index/signature size
Size of leave message	84 bytes(index/PN-A), 80 bytes(CN/PN-S)
Size of update message for index approach	92 Bytes
Size of update message for CN and PN-S	80 bytes
Size of update message for PN-A	84 + size of *change* record
Size of query message	80 + length of search predicate (4 bytes)
Size of response messaege head	88 bytes
Size of result record	4 bytes

- **Message parameters settings:** The message header includes both the Gnutella message header and the TCP/IP header, which is 80 bytes [3]. The size of a search message is the size of message header plus the size of the search predicate (in a network environment with system settings as above, we can simply assume that a search predicate only involves the key attribute, which has a size of 4 bytes by default). In addition to search messages, some other messages are required for maintaining indices and signatures. The size of join messages is the size of message header (80 bytes) plus the size of local index/signatures. For local index and PN-A, the size of leave messages is 84 bytes with 4 bytes of peer identifier and 80 bytes of message header; for CN and PN-S, the size of leave messages is 80 bytes since it is an empty message as explained in Section 3.3. Similarly, the size of update messages for CN and PN-S is also 80 bytes; for PN-A, the size of update message is the size of the message header plus the size of *change* record, also mentioned in Section 3.3. For local index, the size of update messages is 92 bytes with 80 bytes of message header, 4 bytes for node identifier of the peer performing update, 4 bytes to indicate the position of the updated data attribute and 4 bytes for the new key attribute value.

5. Simulation Results

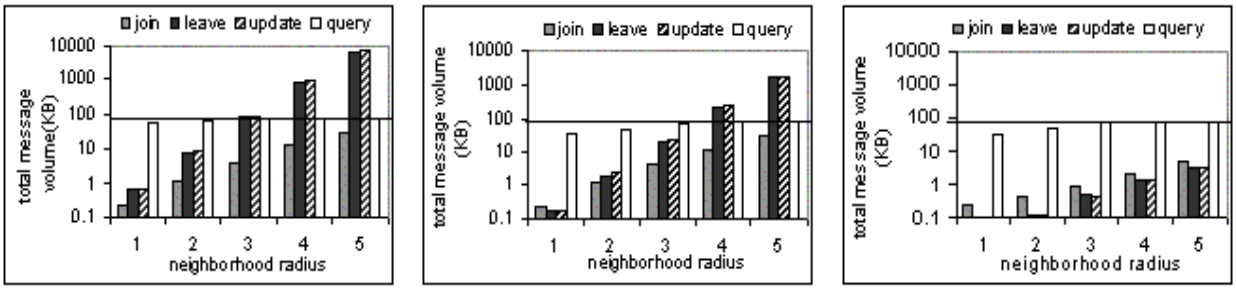
In this section, simulation results for flooding and single-path search are presented. For each of these search methodologies, we present results with both uniform and power-law network topologies. The stop conditions are maximum search depth and minimum number of results for the flooding and single-path search mechanisms, respectively. In the flooding experiments, we compare our signature mechanisms with Gnutella, while in the single-path search we compare with random walk. In both sets of experiments, we also include results for index based search.

5.1. Flooding

The important parameters for investigation include the neighborhood radius for the signature based schemes, size of the key attribute, number of data items, together with the storage size requirements at each node and the ratio of search/update operations for all the mechanisms. In the following experiments, unless explicitly specified, the storage size is set at 1KB and search/update ratio is set at 10, as observed in Gnutella [6].

5.1.1 Uniform network

Neighborhood Radius: Figure 3 shows the total message volume of CN, PN-S and PN-A with different neighborhood radius ranging from 0 to 5 (i.e., the maximum search depth). This is shown in terms of the total message volume of peer join, peer leave, content update and search (query) operations individually. When the neighborhood radius is 0, a peer has no content information about its neighbor peers and the signature approach degenerates to the Gnutella flooding approach in terms of total message volume (this is also denoted by the horizontal line in this figure - which is what we want to compare with). The y-axis is on a logarithmic scale for readability. We can see that the message volume incurred for small neighborhood radius in signature approaches is lower compared to Gnutella flooding. This is due to the desirable (intended) filtering effect achieved by neighborhood signatures. However, when the neighborhood radius is increased, a larger neighborhood information are forced to be compressed within the same space confines (either by superimposing more signatures in CN and PN-S, or appending smaller signatures in PN-A), increasing the false positive probability. This increases the search message volume, though we still find that the search messages volume is not higher than Gnutella even at radius of 5. Of the three signature schemes, the partial signatures are providing better focused searches at smaller radius values. At the same



(a) CN (b) PN-S (c) PN-A

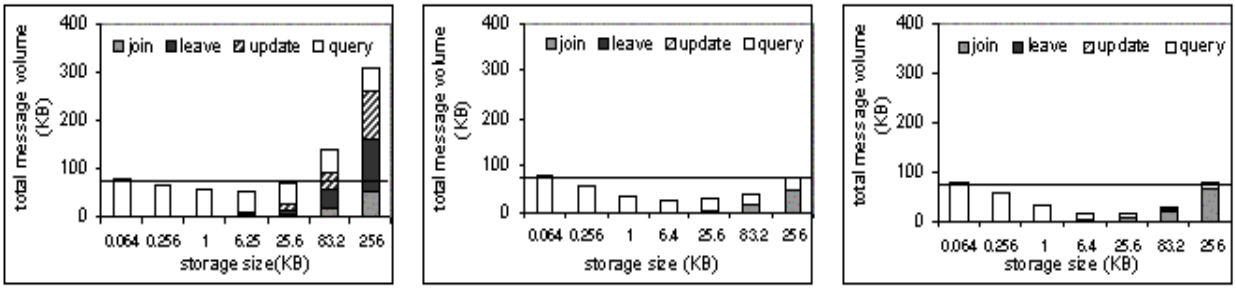
Figure 3. Flooding: effect of neighborhood radius on signature approaches (uniform network). Total message volume of Gnutella is shown by the horizontal line. The y-axis is on logarithmic scale for readability.

time, when the radius is increased, the overheads of peer leave and peer update become much more significant, and even overwhelm the search messages for the superimposing strategies (CN and PN-S). As was mentioned earlier, these schemes require signatures to be generated from scratch. Between these two, the message volume for CN is larger than for PN-S, because only peers along one branch in PN-S are involved for generating new signatures while all neighbors are involved for CN. In contrast to CN and PN-S, the affected signatures can be updated easily in PN-A. Therefore, its leave and update message volume is substantially lower than PN-S and CN.

Table 3. Flooding: optimal neighborhood radius that generates the minimum total message volume for different storage sizes (uniform network).

Storage size(KB)	0.064	0.256	1	6.4	25.6	83.2	256
CN	1	1	1	1	1	1	1
PN-S	1	1	1	1	1	1	1
PN-A	1	1	1	2	3	3	2

Storage Size: Figure 4 shows the total message volume as we allow more storage capabilities at each peer for the signatures. The values shown here use a neighborhood radius that gives the lowest message volume for the given storage size (and these radius values - referred to as optimal radius - for each size are given in Table 3). For CN and PN-S, the optimal radius is 1 for all considered sizes as shown in Table 3. The reason is that when the signature size is small, join/leave/update cost



(a) CN (b) PN-S (c) PN-A

Figure 4. Flooding: effect of storage size on signature approaches (uniform network). Total message volume of Gnutella is shown by the horizontal line.

is small and query cost dominates the total message volume as shown in Figure 4(a) and Figure 4(b). A small neighborhood radius forces less information superimposed together and results in low false positive probability, thereby incurring lower total message volume. When the storage size increases, the cost of join/leave/update dominates the total cost and a smaller neighborhood radius results in lower join/leave/update message volume, providing the best results again. The latter effect (overhead of join/leave/update) is less significant for PN-A, making a higher neighborhood radius more preferable in this scheme.

Size of Key Attribute: Figure 5 shows the total message volume with different sizes of key attribute. In this simulation, we use increased attribute sizes to represent the situations where the (logical) key attribute consists of multiple keys or contain binary data (e.g., music clip). The values shown here uses a given storage size (i.e., 6.4KB) and a fixed number of data items per node (i.e., 400). Thus, by increasing the size of key attribute at a data item from 4 bytes to 1.6KB, the storage/total-attribute-size ratios at a peer for the chosen points in the figure are decreased from $400\%, 100\%, 50\%, 10\%, 5\%$, down to 1%[4]. It can be observed from the figure that the signature approaches outperforms Gnutella and local index significantly as the attribute size becomes large (i.e., the storage/total-attribute-size ratio becomes small). For instance, when the attribute size for each data item is 1.6KB (i.e., storage/total-attribute-size ratio is 1%), the total message volume for PN-A is merely 14% compared to Gnutella and local index. The total message volume for Gnutella increases as the size of key attribute increases, because the search message contains the attribute value(s). Local index performs well when the storage/total-attribute-size ratio is small. However, as the attribute size increases, the given storage size is not sufficient to index all data items[5]. Therefore, local index's performance is the same as Gnutella approach for larger attribute size.

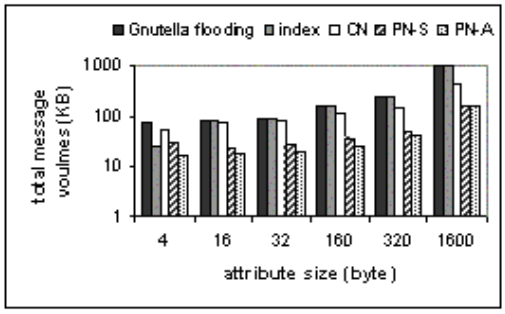


Figure 5. Flooding: effect of key attribute size on signature approaches (uniform network). The y-axis is on logarithmic scale for readability.

[4]The storage/total-attribute-size ratio can be interpreted as the storage overhead corresponding to the key attribute of data items.

[5]The minimum storage overhead for local index is 6.4KB, 25.6KB, 51.2KB, 256KB, 512KB and 2560KB, respectively, for each of the points in Figure 5.

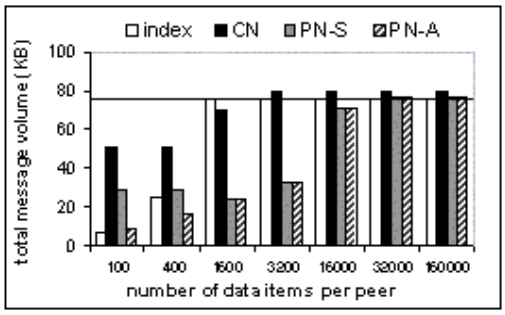


Figure 6. Flooding: effect of number of data items per peer on signature approaches (uniform network). The total message volume of Gnutella is shown by the horizontal line.

Number of Data Items: Figure 6 shows the total message volume as we allow the number of data items per peer to increase from 100 to 160000. With a fixed storage of 6.4KB at each peer, the storage/total-attribute-size ratios for the chosen data points in Figure 6 are $1600\%, 400\%, 100\%, 50\%, 10\%, 5\%, 1\%$, respectively. As shown in the figure, local index outperforms PN-A only when each peer has merely 100 data items (i.e., the storage/total-attribute-size ratio is 1600%). On the other hand, the partial neighborhood signatures performs extremely well as the number of data items per peer increases rapidly. However, when the number of data items is overwhelmed, (e.g., 16000), extra storage size should be allocated for signatures to reduce their false positive probability and the total message volume. Different from the previous figure, the total message volume for Gnutella remains as a constant since the attribute size for each data item is fixed.

Observed from the above two figures, it is obvious that the partial neighborhood signatures are much more storage efficient and flexible than local index. With very little storage overhead, the partial neighborhood signatures can facilitate focused search effectively while local index has some minimal storage requirement.

Search/Update Ratio: Figure 7 shows the optimal total message volume of PN-A (CN and PN-S exhibit similar behavior) with different search/update ratios. We can see that when the storage size is small, the differences are insignificant. However, when the storage size becomes large, the total message volume with lower search/update ratios is significantly higher than that with high search/update ratios due to the high cost of join/leave/update operations at these sizes.

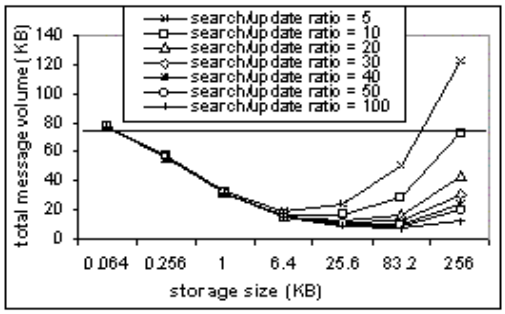


Figure 7. Flooding: effect of search/update_ratio on signature approaches (uniform network). Total message volume of Gnutella is shown by the horizontal line.

Comparing the schemes: Figure 8 compares the performance (messages for all search, join, leave and update operations are lumped together) of the three signature schemes with the index based approach, along with the Gnutella shown as a solid horizontal line. From Figure 8, we can observe that with storage size as small as 256 bytes, the signature schemes can reduce message traffic by over 25% compared to the Gnutella flooding approach. With a higher storage capability, PN-S and PN-A produce even further savings. On the other hand, the local index approach starts outperforming Gnutella only beyond 1KB. At this point, while traffic incurred by index is comparable to Gnutella, CN, PN-S and PN-A incur only 77%, 42% and 42% of

the Gnutella traffic. With a storage size of 6.4KB, all three provide further savings, and so does the local index. However, the message volume of PN-A is only 67% of local index's traffic at this point. As the storage space gets larger, index or signature construction and updates become more expensive (due to join/leave/update operations), causing their message volume to increase again. Even when the storage size keeps increasing, the performance of PN-A is similar to local index. These results demonstrate that the signature approaches (particularly PN-A) can have better performance than the local index with a much smaller storage space requirement.

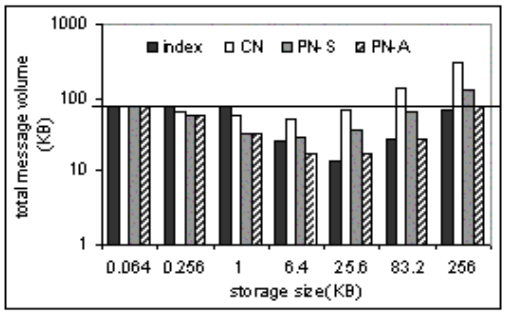


Figure 8. Flooding: total message volume comparison among local index, CN, PN-S and PN-A (uniform network). Total message volume of Gnutella is shown by the horizontal line. The y-axis is on logarithmic scale for readability.

5.1.2 Power-law network

We have performed the same set of experiments as in uniform network topology for power-law networks as well. In the interest of space and clarity, we specifically present results for PN-A that has been consistently shown to be better than CN and PN-S. We also restrict our discussion here to the overall comparison of schemes (shown in Figure 9 for PN-A and local index together with Gnutella flooding), though we have examined the impact of neighborhood radius and search/update ratios as well. We find most of the previous observations are evident here as well, with PN-A doing much better than local index or Gnutella even at 1KB space (nearly half of the traffic generated by local index and Gnutella).

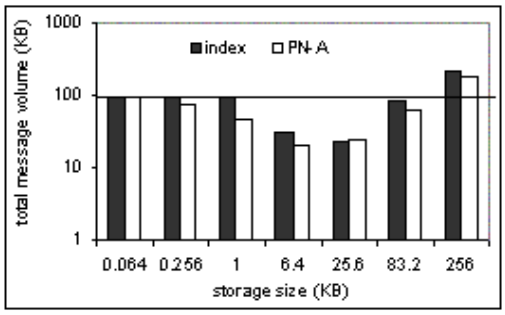
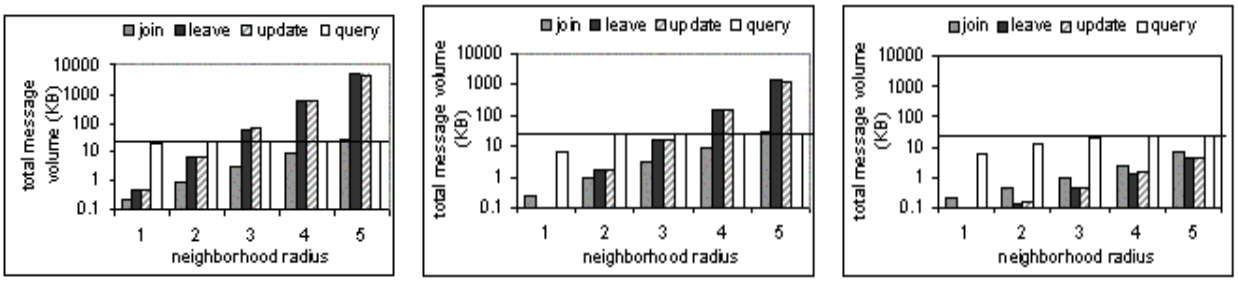


Figure 9. Flooding: total message volume comparison between local index and PN-A (power-law network). Total message volume of Gnutella is shown by the horizontal line. The y-axis is on logarithmic scale for readability.

5.2. Single-Path

In addition to the five parameters (neighborhood radius, storage size, key attribute size, number of data items, and search/update ratio) investigated in flooding search, we include one more parameter - replication ratio - in single-path search since the performance of search with minimum number of results as search stop condition can rely heavily on the number of replicas in the system. Similar to flooding where we compared the schemes with Gnutella and local index, we compare the performance of the proposed signature approaches with random walk and local index. Many of the results are similar to that observed for flooding. They are given here for completeness. Note that, in some figures, the total message volumes for random walk are denoted as a horizontal line.

5.2.1 Uniform network



(a) CN (b) PN-S (c) PN-A

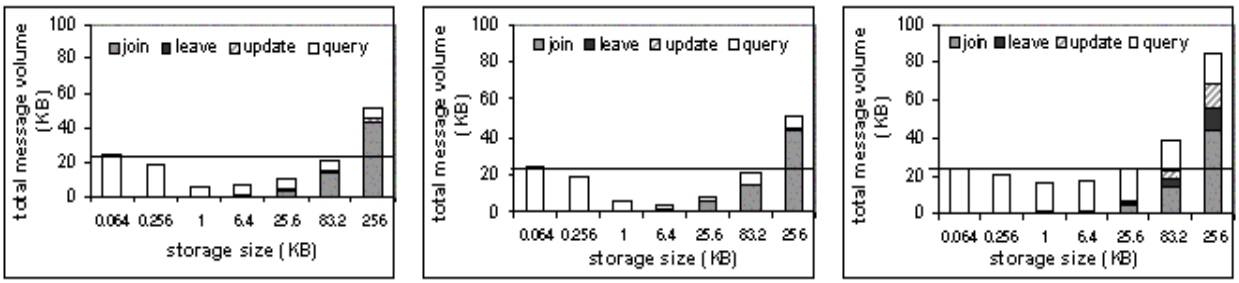
Figure 10. Single-path: effect of neighborhood radius on signature approaches (uniform network). Total message volume of random walk is shown by the horizontal line. The y-axis is on logarithmic scale for readability.

Neighborhood Radius: Figure 10 shows the total message volume of the three signature approaches with different neighborhood radius. Compared to the corresponding figure (Figure 3) in flooding search, the message volume incurred for peer join/leave/update is the same in both cases. This is obvious since the message volume for join/leave/update only depends on the neighborhood radius and is insensitive to the search methods. The trend for search message volume mimics the one for the flooding case. For instance, with the increase of neighborhood radius, the search message volume decreases first due to the filtering effect of neighborhood signature, followed by an increase due to the false positive probability increase when more data are superimposed into a signature with given size.

Table 4. Single-path: optimal neighborhood radius that generates the minimum total message volume for different storage sizes (uniform network).

Storage size(KB)	0.064	0.256	1	6.4	25.6	83.2	256
CN	1	1	1	1	1	1	1
PN-S	1	1	1	1	1	1	1
PN-A	1	1	1	2	2	1	1

Storage Size: Figure 11 shows the optimal total message volume with different storage sizes. Similar to flooding, the values shown here use a neighborhood radius that gives the lowest message volume for the given storage size (and these radius values for each size are given in Table 4). Similar to flooding search, the optimal neighborhood radius for CN and PN-S is 1 due to the high cost of leave/update. For PN-A, the optimal neighborhood radius for small and larger sizes turns out to be 1, and increases to 2 for medium sizes.



(a)CN (b)PN-S (c)PN-A

Figure 11. Single-path: effect of storage size on signature approaches (uniform network). Total message volume of random walk is shown by the horizontal line.

Size of Key Attribute: Figure 12 shows the total message volume with different key attribute sizes. Similar to flooding, the values shown here uses a given storage size (i.e., 6.4KB) and a fixed number of data items per node (i.e., 400). It can be observed from the figure that the signature approaches outperforms random walk and local index significantly as the attribute size becomes larger. The total message volume for PN-A is one order multitude less compared to random and local index when the key attribute size for each data item is 1.6KB (i.e., storage/total-attribute-size ratio is 1%).

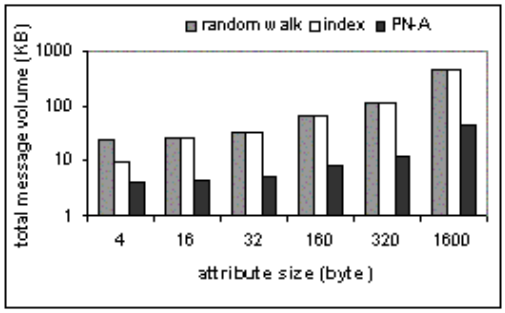


Figure 12. Single-path: effect of key attribute size on signature approaches (uniform network). The y-axis is on logarithmic scale for readability.

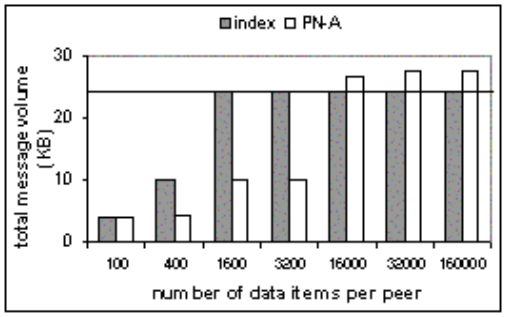


Figure 13. Single-path: effect of number of data items per peer on signature approaches (uniform network). The total message volume of Gnutella is shown by the horizontal line.

Number of Data Items: Figure 13 shows the total message volume as we allow the number of data items per peer to increase from 100 to 160000. The results reconfirm the observations from flooding. The total message volume for local index quickly resorts to ranom walk because the given storage size is not sufficient to index all the data items. The PN-A signature scheme performs very well as the number of data items increases because it needs much less storage to operate effectively.

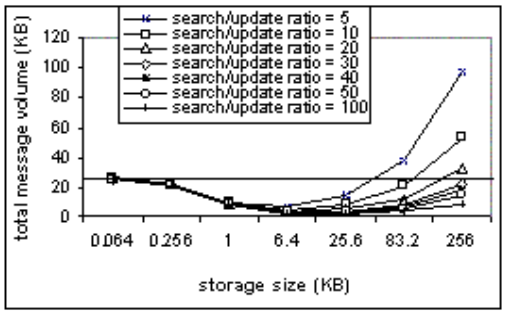


Figure 14. Single-path: effect of search/update ratio on signature approaches (uniform network). Total message volume of random walk is shown by the horizontal line.

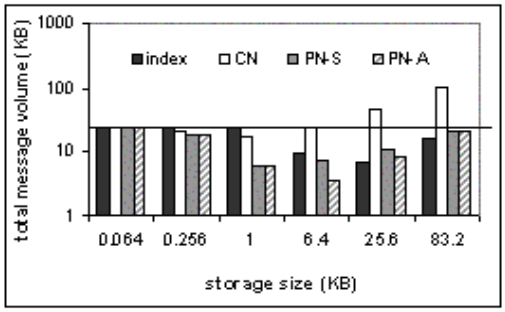


Figure 15. Single-path: total message volume comparison among local index, CN, PN-S and PN-A (uniform network). Total message volume of random walk is shown by the horizontal line. The y-axis is on logarithmic scale for readability.

Search/Update Ratio: Figure 14 illustrates the results with different ratios of search/update operations for PN-A, reconfirming the observations made with flooding.

Comparing the schemes: Figure 15 shows the total message volume comparison between random walk, local index, CN, PN-S and PN-A. Once again, we find the signature schemes (PN-A in particular) are able to incur lower message traffic in retrieving the required number of data items at a much lower storage cost than local index.

Replication Ratio: Figure 16 compares random walk, local index and PN-A for different degrees of replication of a data item in the network. In these experiments, both local index and PN-A are run with a storage size of 6.4KB (local index only starts to provide reasonable performance with storage size 6.4KB). At high degrees of replication, we expect random walk to do rather well, since there is a higher likelihood of finding the necessary number of data items even when randomly traversing the network (without incurring any join/leave/update overheads). However, at lower degrees of replication it does much worse than the signature or indexing approaches which can direct searches in a more focussed manner. Of these two, we find that PN-A is more effective at reducing traffic even at very small degrees of replication. PN-A incurs an order of magnitude lower message traffic with respect to random walk for a replication ratio of 0.1% and only 17% of random walk traffic with a replication ratio of 0.5%. Compared to local index, PN-A incurs 29% of index traffic for a replication ratio 0.1% and 43% of index traffic for a replication ratio 0.5%.

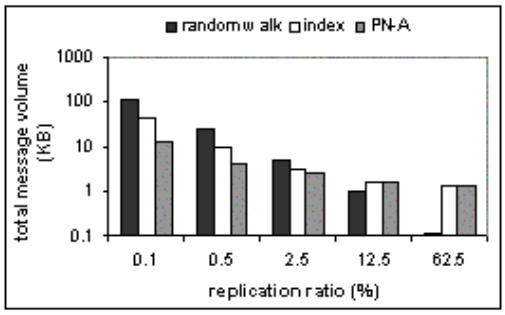


Figure 16. Single-path: effect of replication ratio on random walk, local index and signature approaches (uniform network topology). The y-axis is on logarithmic scale for readability.

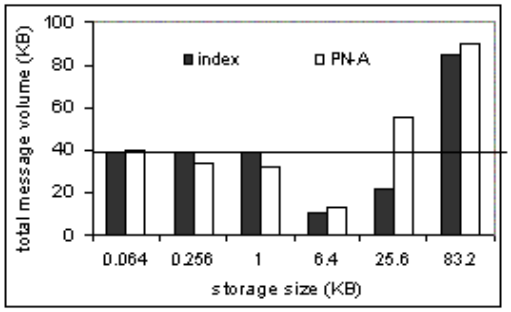


Figure 17. Single-path: comparison among random walk, local index and PN-A with different storage size (power law network). Total message volume of random walk is shown by the horizontal line. The y-axis is on logarithmic scale for readability.

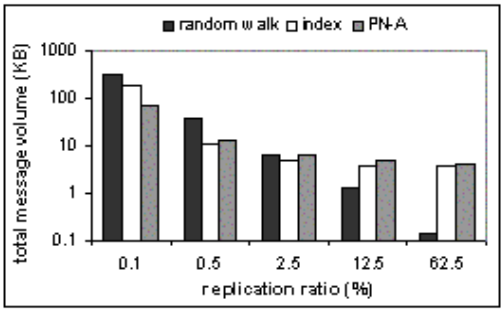


Figure 18. Single-path: effects of replication ratio on random walk, local index and PN-A (power law network). The y-axis is on logarithmic scale for readability.

5.2.2 Power-law network

We again focus on PN-A which gives the best performance among the signature approaches, and compare it with random walk and local index for different storage sizes in Figure 17. A comparison of these schemes with different degrees of replication is shown in Figure 18. These results reconfirm the benefits of PN-A over local index and random walk, showing

how it can incur fewer messages to retrieve the required number of data items at a smaller storage cost than local index and when the number of replicas is not very high (where random walk suffices).

6. Concluding Remarks and Future Work

Peer-to-Peer (P2P) applications such as Napster and Gnutella have made the Internet a popular medium for resource and information exchange between thousands of participating users. A primary consideration in the design of such applications is the high network traffic that they can generate when searching for resources/information. Existing research work has shown that maintaining auxiliary information at the network nodes can lead to more focussed searches, reducing network traffic considerably. One can argue that with infinite storage capacity it is possible to maintain complete auxiliary information on all the resources of a P2P network at a peer node, leading to extremely efficient searches. However, this not only incurs high storage overheads, but also additional costs - to update the auxiliary information - when nodes join/leave the system or when information is updated. This trade-off between storage space vs. network traffic opens up a rich space of mechanisms to explore. Previous research has looked into one possible mechanism - index based search - in this space. We have proposed three new mechanisms, based on signature files from information retrieval, within this space that can provide a better focussed search at a lower storage overhead than local index.

When storage is not an option, or when search operations are not that compelling, schemes such as Gnutella (which floods the network) or Random Walk (which randomly traverses one node after another) serve their purpose reasonably well. However, maintain auxiliary information about the neighboring peers can considerably improve performance. We have shown that at a much smaller storage cost, relative to the overall space consumed by the data items themselves, signatures are quite effective at reducing search costs, compared to index. At these storage requirements, the overheads of join/leave/update operation are also adequately compensated by the savings in search messages. Of the three schemes - CN, PN-S and PN-A - that we proposed, PN-A gives the best performance.

The schemes have been extensively evaluated both with the intention of fine-tuning parameters that they use (the neighborhood radius, storage size) and to perform the comparison with the previous proposals for different network topologies (uniform and power-law), different proportions of operations (search/update), different sizes of key attribute, different number of data items at a peer, different degrees of data replication, different search stop conditions and with both flooding and single-path search strategies. We uniformly find PN-A giving good savings in message volume over Gnutella, random walk and local index approaches at a small storage cost. In addition to the performance and storage savings with signatures, there are a couple of other advantages that they exhibit compared to index-based approaches: (a) Unlike index, they can search across multiple attributes by appropriately encoding all the attributes when composing them, instead of being restricted to one or a small number of attributes which needs to be predetermined. This facilitates keyword and content based search, etc. (b) It takes a certain minimum amount (threshold) of storage to compute (and store) an index. Any fraction of this threshold does not help this scheme, and we have to resort to broadcasts/flooding. On the other hand, signatures do not impose any such restrictions and can work with any amount of space allotted to them (though when the space gets too small the ability to focus the search diminishes, and in the worst case false positives can lead to flooding). All these observations lead us to believe that PN-A is an extremely popular mechanism for implementing resource and information lookup operations in P2P networks.

Our ongoing work is looking into reducing false positive effects in signatures by exploiting real data patterns. We are also looking into improving leave and other operation overheads, together with incorporating with other optimizations such as intermediate node caching and peer clustering. Finally, we are investigating P2P applications overlayed on wireless networks.

References

[1] Clips company. http://www.clip2.com/gnutella.html.

[2] Freenet website. http://www.freenet.com.

[3] Gnutella website. http://gnutella.wego.com.

[4] Morhpeus website. http://www.musiccity.com.

[5] Napster website. http://www.napster.com.

[6] DSS Group, Gnutella: to the bandwidth barrier and beyond. http://dss.clip2.com, June 2000.

[7] K. Aberer. P-Grid: a self-organizing access structure for P2P information systems. In *Sixth International Conference on Cooperative Information Systems (CoopIS)*, pages 179–194, Trento, Italy, 2001.

[8] K. Aberer, M. Punceva, M. Hauswirth, and R. Schmidt. Improving data access in P2P systems. *IEEE Internet Computing*, 6(1):58–67, 2002.

[9] L. A. Adamic, R. M. Lukose, A. R. Puniyani, and B. A. Huberman. Search in power-law networks. *Physics Review E*, 64:46135–46143, 2001.

[10] R. Bayer. Symmetric binary B-trees: Data structure and maintenance algorithms. *Acta Informatica*, 1(4):290–306, 1972.

[11] D. Comer. The ubiquitous B-tree. *ACM Computing Survey.*, 11(2):121–137, 1979.

[12] A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer systems. In *Proceedings Of the 22 nd IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 23–34, July 2002.

[13] F. M. Cuenca-Acuna, R. P. Martin, and T. D. Nguyen. PlanetP: using gossiping and random replication to support reliable peer-to-peer content search and retrieval. Technical Report DCS-TR-494, Rutgers Universit, July 2002.

[14] C. Faloutsos and D. W. Oard. A survey of information retrieval and filtering methods. Technical Report CS-TR-3514, University of Maryland, August 1995.

[15] M. A. Jovanovic, F. S. Annexstein, and K. A. Berman. Modeling peer-to-peer network topologies through "small-world" models and power laws. In *Telecommunications Forum (TELFOR)*, November 2001.

[16] J. Kubiatowicz et al. Oceanstore: An architecture for global-scale persistent storage. In *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 190–201, Novermber 2000.

[17] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *Proceedings Of the 16th ACM International Conference on Supercomputing*, pages 84–95, June 2002.

[18] S. Ratnasamy, P. Francis, M. Handley, R. M. Karp, and S. Schenker. A scallable content-addressable network. In *Proceedings of ACM SIGCOMM*, pages 161–172, August 2001.

[19] A. I. T. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings Of the 18th IFIP/ACM International Conference on Distributed Systems Platforms(Middleware 2001)*, pages 329–350, November 2001.

[20] K. Sripanidkulchai. The popularity of gnutella queries and its implications on scalability. http://www-2.cs.cmu.edu/~kunwadee/research/p2p/gnutella.html, Feburary 2001.

[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*, pages 149–160, August 2001.

[22] B. Yang and H. Garcia-Molina. Designing a super-peer network. Technical Report 2002-13, Stanford University, February 2002.

[23] B. Yang and H. Garcia-Molina. Improving search in peer-to-peer networks. In *Proc. Of the 22 nd IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 5–14, July 2002.

[24] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: an infrastructure for fault-tolerant wide-area location and routing. Technical Report UCS/CSD-01-1141, Computer Science Division, U. C. Berkeley, April 2001.