Performance Evaluation of Neighborhood Signature Techniques for Peer-to-Peer Search

Mei Li Wang-Chien Lee[*] Anand Sivasubramaniam

Department of Computer Science and Engineering

Pennsylvania State University

University Park, PA 16802, USA

E-Mail: {meli, wlee, anand}@cse.psu.edu

Received June 13, 2007 ; Accepted June 30, 2007

Abstract. Peer-to-peer (P2P) systems have received a lot of attention due to the popularity of applications such as SETI, Napster, Gnutella, and Morpheus. The P2P systems present tremendous challenges in searching data items among the numerous host nodes. While search has been studied in a similar but different context, i.e., parallel and distributed database systems, the large scale and dynamic membership change of P2P systems require the search issue to be re-examined. Existing search techniques in unstructured peer-to-peer overlay networks incur excessive network traffic. In this paper, we investigate the issues of trading-off storage space at peers to reduce network overhead in unstructured P2P overlay networks. We propose to use signatures for directing searches, and introduce three schemes, namely complete-neighborhood signature (CN), partial-neighborhood superimposed signature (PN-S), and partial-neighborhood appended signature (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 analysis and simulations 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 the best performance at a small storage cost.

Keywords: signature, index techniques, information search, peer-to-peer systems, performance evaluation

1 Introduction

The advance of facilities such as Napster [1] and Gnutella [2] 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, host nodes in these Peer-to-Peer (P2P) systems can act as servers as well as clients. Despite avoiding performance bottlenecks and single points of failure, the P2P systems present tremendous challenges in searching data items among these numerous host nodes.

Search of information in parallel and distributed database systems has received a lot of research efforts from the database community (please see [3] for a comprehensive survey). While peer-to-peer systems are similar to shared-nothing parallel and distributed database systems, the challenges faced are fundamentally different. Parallel and distributed database systems consist of dozens or hundreds of machines that are designated to provide certain database services, and thus are rather stable. In contrast, P2P systems may have nodes join or leave frequently, and thus are highly dynamic. In addition, the size of a P2P system is in the range of thousands or even millions of nodes, which is much greater than the typical sizes of parallel and distributed database systems. Thus, P2P systems require the search techniques to tolerate membership changes (i.e., node join, leave, or failure), and to scale to a large number of nodes, in addition to locating data items efficiently. As a result, the search techniques developed for parallel and distributed database systems can not be simply employed to peer-to-peer systems and the search issue needs to be re-examined.

Existing P2P overlays can be classified as unstructured P2P overlays and structured P2P overlays. The main differences between these two are whether data placement and network topology are controlled. In a unstructured P2P overlay, like Gnutella [2], peers form a random topology and store data items locally. Primary search techniques proposed for unstructured P2P overlays are flooding and random walk [4,5]. While the search costs in unstructured P2P overlays may not be low in terms of the total number of messages and/or the number of hops traversed per search, the advantages are in the low maintenance cost, making them relatively easy to handle

[*] Correspondence author

membership and data content changes. In addition, unstructured P2P overlays pose no restrictions on the types of queries that can be supported effectively. In contrast, structured P2P overlays tightly control data placement and network topology to perform certain kind of search ordering which can facilitate searching for the requested data items. CAN [6], Chord [7], Pastry [8], Tapestry [9], and SSW [10] are examples of structured P2P overlays. Search over these overlays is efficient (search path length is O(*logN*) where *N* is the network size). However, they incur high overheads for data placement and topology maintenance. Most of existing systems deployed in practice are unstructured (for the simplicity and flexibility). In this study, we focus on improving search efficiency of unstructured P2P overlays. In the rest of this paper, we refer unstructured P2P overlays as P2P overlays for simplicity.

A P2P network[1] is established by logical connections among the participating nodes (called peers). The peers provide digital information resources, such as music clips, images, documents and other forms of digital content, to be shared with other peers. The P2P overlay network topology may change dynamically due to constant joins and leaves of the peers, namely peer join and peer leave[2], in the network. In addition, the shared information changes dynamically since the peers may update the digital content that they offer, namely peer update.

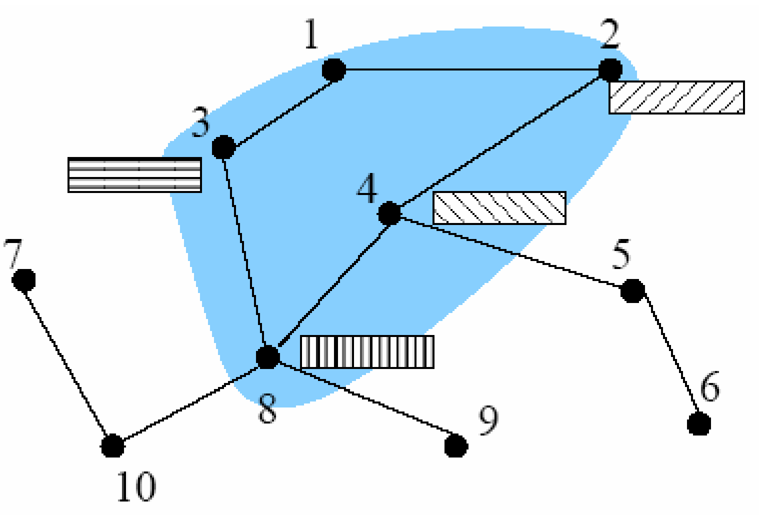


Fig. 1. A partial snapshot of a P2P network

Fig. 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 P2P overlay 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 a sequence 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. Each peer has a *neighborhood*, which includes all the peers reachable within a given distance. The *neighborhood radius* refers to the distance from a peer to the edge of its neighborhood. For example, as illustrated in Fig. 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. Node 1 has a neighborhood of radius 2 that consists of Nodes 2, 3, 4, 8.

Two main strategies have been explored for searching in unstructured P2P overlays:

- **Blind search:** This strategy lets messages poll nodes, without having any idea of where the data may be held, till the required items are found. Gnutella and random walk use such a strategy. In Gnutella, a search message is forwarded by a peer to all its neighbors until the message reaches a certain preset distance. The down side of this strategy is the possible network overload due to a large number of generated search messages. 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 randomly selected neighbors. However, this approach incurs a long latency to satisfy a request.

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

[2]Peer failure can be treated similarly as peer leave as discussed in Section 2.4.2. Thus, we do not list it separately.

- **Directed search:** This strategy maintains additional information in the peer nodes (which blind search does not require) in order to reduce network traffic. Consequently, messages are directed specifically along paths that are expected to be more productive. The additional information is typically maintained as indexes over the data that are contained either within hierarchical clusters [11] or by nearby neighbors [12, 13, 14]. In addition to the high storage cost incurred by storing the index itself and high maintenance overhead incurred by index update, this indexing approach requires determining what attributes to index a priori, thus constraining the search that can be supported.

While index based directed search seems attractive in terms of search message traffic, the drawbacks as described above motivate us to apply *signature* techniques to provide flexible search ability (i.e., supporting arbitrary queries) and better search message traffic behavior at a lower storage cost and maintenance overhead than index-based mechanisms.

Signature methods have been used extensively for text retrieval, image database, multimedia database, and other conventional database systems. A signature is basically an abstraction of the information stored in a record or a file. By examining the signature only, we can estimate whether the record contains the desired information. Due to its compactness, signature incurs low storage as well as low communication overheads when being exchanged among remote hosts. In addition, signature can support arbitrary queries. All these advantages of signature make it very suitable for filtering information stored at nodes of P2P systems. This paper presents three novel ways of using signatures, namely *complete neighborhood signature (CN), partial neighborhood superimposed signature (PN-S) and partial neighborhood appended signature (PN-A)*, to represent neighborhood data at network nodes for optimizing searches in P2P systems.

The merits of these three neighborhood signature schemes are exploited by analysis and simulations. Since the signature techniques trade some extra storage overhead for reducing network traffic[3], we use total message volume to evaluate various P2P search techniques. We derive an analytic model to estimate the search cost and the maintenance overhead of the proposed signature schemes and conduct an extensive performance evaluation through both analysis and simulation to compare their performance with some representative search techniques developed for unstructured P2P overlays, including Gnutella [2] and random walk [5] for blind search and local index [14] for directed search. We examine the performance of these techniques under different network topologies (i.e., uniform and power-law networks) and searching strategies (i.e., flooding and single-path), and test their sensitivity to various factors including neighborhood radius, storage constraints, key attribute size, number of data items at a node, data distribution, etc. Our experiments show that the signature approaches (particularly PN-A) are much better than the other alternatives for the most reasonable storage space availability assumptions on host nodes.

The basic idea of the three neighborhood signature techniques have been discussed in our preliminary work [15]. In this paper, we provide a complete presentation of various operations using these neighborhood signatures. In addition, we propose a lazy signature update method that improves the signature maintenance overheads. Moreover, we provide an analytic model and conduct an in-depth performance evaluation using both analytic and simulation experiments. The main contributions of this paper are three-fold:

- Three novel neighborhood signature schemes for efficient search in P2P networks are proposed. The details for various operations, such as search, peer join, leave and update, are presented.
- An analytic model to estimate the search cost and maintenance overhead of the neighborhood signature schemes is derived.
- An extensive performance evaluation using both analysis and simulation to compare our proposal with other existing P2P searching approaches is conducted. To the best of our knowledge, this is the first study that takes both of search cost and maintenance overhead into in-depth consideration.

There have been techniques suggested to store additional information on intermediate nodes, e.g., cache query results [16] or maintain history about prior operations [14], to reduce network traffic. While the effectiveness of such enhancements depends 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 [17]. This service uses a technique called *bloom filte*r, which is similar to the signature technique used in this paper, to efficiently summarize the indexed terms. However, its focus is on providing a search engine rather than reducing the network traffic.

The rest of the paper is organized as follows. In Section 2, we present details of the proposed neighborhood signature schemes. In Section 3, we provide a qualitative comparison between our proposal and prior searching approaches. In Section 4, we present an analytic model for various costs incurred by search and maintenance of

[3]Signature techniques reduce search latency as well. As we will discuss in Section 3, the improvement on search latency can be easily observed, and thus we focus on the improvement on network traffic in the paper.

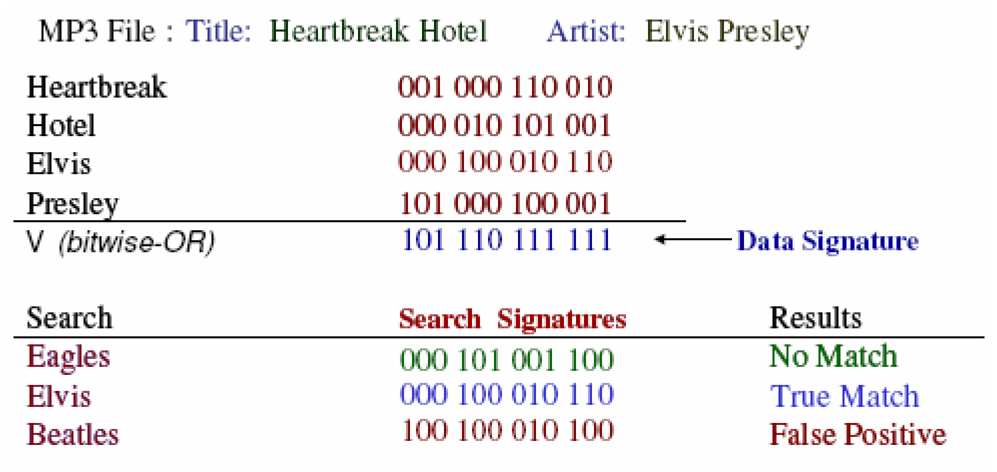
our proposed schemes. Section 5 gives the experimental setup for the performance evaluation and detailed results from experiments under different search strategies. Finally, Section 6 summarizes the contributions of this paper and outlines directions for future work.

2 Neighborhood Signatures

In this section, we first provide some background on the signature method and then extend it for search in P2P networks. We propose three neighborhood signature schemes, CN, PN-S, and PN-A, to index the data content offered within the neighborhood of a peer, which help to direct a search to a subset of probabilistically productive neighbors. We describe the formation of the signatures for each scheme and then provide details for search and signature maintenance under various scenarios (i.e., peer join, peer leave, and peer update).

2.1 Background

Signature techniques 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 performing a bitwise-OR operation to superimpose (denoted by □) them together. Fig. 2 depicts the signature generation and comparison processes of a digital file and some searches



MP3 File : Title: Heartbreak Hotel Artist: Elvis Presley

Heartbreak	001 000 110 010	
Hotel	000 010 101 001	
Elvis	000 100 010 110	
Presley	101 000 100 001	
V *(bitwise-OR)*	101 110 111 111	← Data Signature

Search	Search Signatures	Results
Eagles	000 101 001 100	No Match
Elvis	000 100 010 110	True Match
Beatles	100 100 010 100	False Positive

Fig. 2. Illustration of signature generation and comparison

As illustrated in the figure, 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 a match (i.e., for some bit set in the search signature, the corresponding bit in the data signature is NOT set), the corresponding document can be ignored. Otherwise (i.e., for every bit set in the search signature, the corresponding bit in the data signature is also set), there are two possible cases: 1) true match - The document is indeed what the search is looking for. 2) *false positive* - Even though the bits of the signatures may match, the document itself does not match the search criteria. This occurs due to certain combinations of bit strings generated from various attribute values, keywords, or document content. The storage devoted to the signature can influence the probability of false positives[4]. Obviously the documents with matching signatures still need to be checked against the search criteria to distinguish a true match from a false positive.

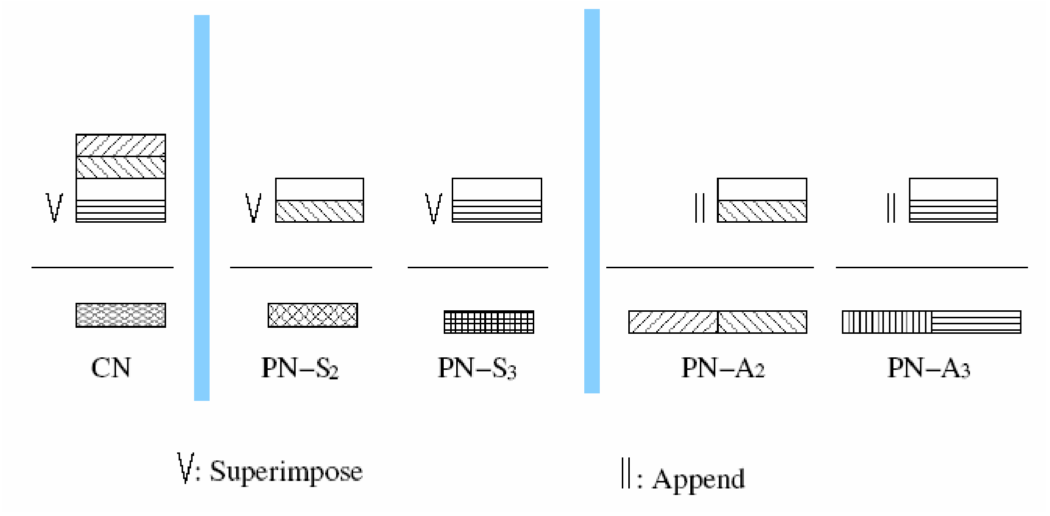
2.2 Signature Schemes for P2P Systems

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 at 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 digital content available within a specific network distance (i.e., its neighborhood). Therefore, a peer

[4]The formula for estimating the probability of false positives is presented in Section 4.

can 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. Fig. 3(a) shows an example of a complete neighborhood signature for peer 1, which indexes all the content available at Peers 2, 3, 4, and 8, whose local signatures are represented by rectangles with different filling patterns in Fig. 1. 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) CN (b) PN-S (c) PN-A

Fig. 3. Illustration of neighborhood signature generation

- **Partial Neighborhood (PN):** While the CN scheme has the advantage of jumping out of a neighborhood when the search signature and neighborhood signatures do not match, it has to forward the search to all of its neighbors when there is a match between the search signature and neighborhood 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, which is called *partial neighborhood signature*. 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 of the local signatures located within a neighborhood branch are compressed into one signature, called *partial neighborhood superimposed (PN-S) signature*. Fig. 3(b) shows that peer 1 has 2 PN-S signatures, where PN-S_2, the neighborhood signature for branch 2, indexes all the contents available at Peers 2 and 4, and PN-S_3, the neighborhood signature for branch 3, indexes the contents available at Peers 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 a clue of which peers should be visited, resulting in searching all of these peers. An alternative that we propose, called *partial neighborhood appended (PN-A) Signature*, is to append (concatenate) all of the local signatures within a branch of the neighborhood into a partial neighborhood signature[5]. When a search signature matches with some sub-signatures within a PN-A signature, the search message will only be forwarded to these peers associated with the matched sub-signatures. Fig. 3(c) shows that peer 1 has 2 PN-A signatures, where PN-A_2 indexes all the contents available at Peers 2 and 4, and PN-A_3 indexes the contents available at Peers 3 and 8.

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

- 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.

2.3 Search Algorithms

The neighborhood signature schemes are generic mechanisms that can adapt to different search strategies and protocols. A user may initiate a search of digital content from any peer in the network, and the search message is forwarded to all or a subset of its neighbors to extend the search. In order to prevent indefinite search message propagation in the P2P network, a stop condition needs to be specified in the query. 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 the following, we discuss the signature based search algorithms for the following two strategies: flooding/maximum-depth and single-path/minimum-result. For clarity of our presentation, we use r to denote the radius of a neighborhood.

2.3.1 Flooding Search

In this section, we describe how a peer utilizes neighborhood signatures to perform searches with maximum search depth as the stop condition. 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.

Algorithm 1 Flooding search based on complete neighborhood signatures

```
Incoming Message: Search_Msg(TTL)
Local Variables: Local_Sig, Neighborhood_Sig, Search_Sig
System Parameters: r {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 sender.
7:    end if
8: end if
9: {check whether the maximum search depth is reached}
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 $TTL > r$ then
18:       forward the message Search_Msg($TTL - r - 1$) to all the neighbors located $r + 1$ hops away.
19:    end if
20: end if
```

Algorithm 1 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 its local signature. If there is a match, the content at this peer node is examined to determine whether this is a true match or a false positive. If this is a true match, a pointer to the result is returned back to the sender. Next, the peer checks the *TTL* to see whether the maximum search depth 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 of 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 *r* hops (the neighborhood) need not be checked (as a result, the search message is dropped when $TTL \leq r$). In this case, the search should be processed only at the peers r + 1 hops away[6] .

The flooding search algorithms for the two partial neighborhood (PN) signature schemes are only slightly different from the one discussed above (refer to Lines 14-19). Instead of maintaining only one neighborhood signature for the whole neighborhood, a partial neighborhood signature is generated for each of the neighbor branches in these two schemes. The PN-S scheme compresses all of the local signatures in a neighborhood branch into one signature (by superimposing), while the PN-A scheme enlists (appends) all of 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 $r + 1$ 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 examining all of 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, similar to PN-S, the search message is forward to the peers $r + 1$ hops away, located right outside of the partial neighborhood corresponding to this neighborhood signature.

2.3.2 Single-Path Search

In this section, we describe how a peer utilizes neighborhood signatures to perform single-path search with 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, one peer instead of all peers located $r + 1$ 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. Each time a result is found, the *TNR* is increased by 1. The search is stopped when *TNR* reaches a pre-defined value.

For CN signature, if the neighborhood signature matches with the search signature, all peers in the neighborhood 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*, refer to Lines 15-19). Different from CN, the neighborhood checking messages are only forwarded to the neighbors with matched neighborhood signatures in PN-S, or directly to the peers with matched 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: r {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 sender.
8:     end if
9: end if
10: {check whether enough results are found}
11: if $TNR \geq R$ 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, r) to all the neighbors.
17:     wait for a T period of time to receive replies and forward result pointers back to the sender.
18:     increase TNR by the number of received result pointers.
```

[6] Here we assume that a peer has the knowledge of its peers at $r+1$ hops away so that it may forward the search messages directly. This knowledge can be simply obtained through its 1-hop-away neighbors since the peers at r hops away from these 1-hop-away neighbors are exactly the peers at $r + 1$ hops away.

```
19: end if
20: if $TNR$ < R then
21:     forward Search_Msg($TNR$) to a randomly selected peer located $r + 1$ hops away.
22: end if
```

2.4 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 signatures 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. In the following, we adopt two different strategies to conduct signature update. One update strategy is *Eager Update* where a newly joined peer (a leaving peer, or a peer conducting content update) proactively informs other peers located within its neighborhood to update the affected neighborhood signatures immediately. The second update strategy is *Lazy Update* where the updates on the neighborhood signatures are postponed till necessary, i.e., a search is encountered. In the following, we first describe the actions to be taken at peer join, peer leave, and peer update for eager update strategy. We then outline the difference between eager update and lazy update strategies.

2.4.1 Eager Update on Signatures

- **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 included 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 its neighborhood signatures. Besides this, some peers that were not in the same neighborhood earlier, may be brought into a 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 local signatures from the peers in the neighborhood by sending to these peers a *pseudo join messag*e. The pseudo join message functions similarly 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 neighborhood signature. Slightly different from CN, for PN-S, the affected peers only need to ask for individual local signatures from the peers on the affected branch.
- **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. For PN-A, the difference between the updated signature and the old signature is recorded in a change record, which is included in the update message, so that the affected peers can update the relative sub-signatures in their neighborhood signatures accordingly.

2.4.2 Lazy Update on Signatures

In lazy update, the signature update is postponed till a search is encountered. However, a newly joined peer or a peer conducting content update needs to notify the other peers in the neighborhood because their affected neighborhood signatures have become stale. With this notification, a peer in the neighborhood can invoke signature update only when a search is encountered. This notification is not necessary for peer leave since using a stale neighborhood signature which includes information about data items stored in the leaving peer will not result in missing of any data items stored in the system.

- **Peer join:** To perform the notification as described above, a newly joined peer sends a simple message with its node identifier (instead of its local signature as in eager update) to the peers in its neighborhood. A peer in the neighborhood records the received node identifier in a *To-Join-Lis*t. If a peer with a non-empty

To-Join-List receives a search message later on, it invokes actions similar to peer join in eager update (as described above). During this process, the peers that have updated their neighborhood signatures affected by this peer join remove the relevant entry from their To-Join-Lists, accordingly.

- **Peer leave:** When a peer leaves the network, actions on signature maintenance are not taken immediately. Later on, during search process, when a peer detects that a neighboring peer has left the network, it invokes actions similar to peer leave in eager update (as described above).
- **Peer update:** Similar to peer join, a peer updating its local content sends a message with its node identifier to the peers in its neighborhood, which then record the received node identifier in their *To-Update-List*. When one of these peers receives a search message, it invokes actions similar to peer update in easy update (as described above). During this process, the peers that have updated their neighborhood signatures remove the relevant entry from their To-Update-List, accordingly.

3 Qualitative Comparison of Different Search Schemes

Before a quantitative evaluation, we first provide a qualitative comparison of our proposal with existing representative P2P search techniques developed for unstructured P2P systems, such as Gnutella, random walk, and local index (see Table 1).

Gnutella incurs high search message volume since each search message is flooded in the network. Random walk only forwards a search message from a peer to one or a few of its neighbors, thus the search message volume is reduced compared to Gnutella flooding approach. Local index and signature approaches incur low search message volume since the search is only forwarded to a subset of peers in a neighborhood being searched. In addition to the search message volume, both local index and signature approaches incur message exchanges for index/signature construction and maintenance during peer join/leave/update, which Gnutella and random walk do not incur. This message volume overhead incurred by peer join/leave/update is proportional to the index/signature size. Since the index size is much larger than signature size, the maintenance overhead for local index is usually larger than for signatures. In addition, this overhead can be offset when search is much more frequent. The trade-off between these is investigated quantitatively in the rest of this paper.

Table 1. Comparison between Gnutella flooding, random walk, local index and signature approaches

	Gnutella	Random Walk	Local Index	Signature
Search cost	high	moderate	low	low
Maintenance cost	none	none	moderate	low-moderate
Search latency in flooding search	low	-	low	low
Search latency in single-path search	-	high	moderate	moderate
Storage requirement	none	none	high	low
Search terms	arbitrary	arbitrary	key attributes	arbitrary

In Gnutella, the search latency is proportional to the search radius, which are about 5 to 7 hops. Random walk's search latency is inversely proportional to the number of replicas in the network as described in [5]. When the number of replicas is low in the network, the search latency for random walk would be very high compared to Gnutella. In flooding search, the search latency of local index and signature approaches is comparable to Gnutella. In single-path search, the search latency for local index and signature approaches is reduced approximately S_r times (S_r is the size of the indexed neighborhood with radius r) comparing to random walk. Since the search latency can be easily observed as described above, we didn't include plots for search latency in the performance evaluation section.

Both local index and signatures improve performance at the expense of extra storage. However, the storage requirement of signature is much lower than local index. Moreover, the signature schemes can adapt to the available storage. In addition to incurring lower 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, local index approach is not a good choice since it only supports searches against some selected key attributes.

4 Analysis of Signature Approaches

In this section, we develop an analytical model for the search and maintenance cost of the proposed neighborhood signatures under uniform network topology. The analysis considers both of flooding and single-path search strategies. Table 2 lists the symbols used in this section for easy references. Since most terms in the table are self-explanatory, we only explain some of them in the following. 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 redundant edges. The ratio of redundant edges in the network is denoted as β. *Replication ratio*, denoted as α, represents the number of replicas per data divided by the total number of peers in the network. *Search/update ratio*, denoted as φ, indicates the relative proportion of search operations to the other operations that require signature maintenance (i.e., peer join, peer leave and peer update).

Table 2. Symbols used in the analysis.

System parameters	
N: number of nodes in the network	*B*: number of neighbors per node
d: number of data items per node	α : replication ratio
β : ratio of redundant edges	φ : ratio of search to peer join/leave/update
Stop conditions	
T: maximum search depth	*R*: minimum number of results
Index/signature parameters	
r: neighborhood radius	P_i : number of nodes at hop distance i
S_i : size of a neighborhood with radius i	
Signature parameters	
Fd : false positive probability	μ: signature match ratio
Message parameters	
M_h : size of message header	M_j : size of join message
M_l : size of leave message	M_u : size of update message
M_s : size of search message	M_r : size of response message header
s: size of response record	
Accounting parameters	
V_{search} : message volume incurred by search forwarding	$V_{response}$: message volume incurred by search response
V_{join} : message volume incurred by peer join	V_{leave} : message volume incurred by peer leave
V_{update} : message volume incurred by peer update	V_{total} : total message volume
Q_i : number of search messages forwarded to nodes i hops away	

We present the formulae for message volume incurred by search, peer join, peer leave and content update, respectively. The total message volume is simply a weighted summation of the message volume incurred by various operations.

$$V_{total} = \frac{V_{join} + V_{leave} + V_{update}}{\varphi} + V_{search} + V_{response}$$

where the notations of message volumes corresponding to various operations are self-explained (also see Table 2).

Before we proceed to derive formulae for those message volumes, we first obtain the number of nodes at hop distance i (denoted as P_i), the size of a neighborhood with radius j (denoted as S_j), and signature match ratio (denoted as μ), since these variables will be used extensively later on.

Let the number of neighboring peers of a peer node at one hop distance be *b*. Each of these neighboring peers has $b-1$ connections with 2-hop-away peers. Assuming β portion of the connections is redundant, the number of nodes at 2-hop distance is $b(b-1)(1-\beta)$. Thus, the number of nodes at hop distance *i* is derived as follows:

$$P_i = b(b-1)^{i-1}(1-\beta)^{i-1}.$$

The size of a neighborhood with radius *j* can be derived from P_i as follows:

$$S_j = \sum_{i=1}^{j} P_i .$$

Matched signatures include true matches as well as false positives, thus, μ can be expressed as

$$\mu = \frac{true\_match + false\_positive}{target\_size} \tag{1}$$

where *target_size, true_match* and *false_positive* are the number of signatures to be examined, the number of true matches and the number of false positives, respectively. According to [18], the mathematical definition of *false positive probability*, Fd, is

$$Fd = \frac{false\_positive}{target\_size - true\_match}. \tag{2}$$

The signature match ratio, μ, can thus be expressed as $Fd + (1-Fd)\cdot$ *true_match/target_size* based on Equations (1) and (2).

In complete neighborhood (CN) signature scheme, the value of *true_match* can be approximated as the product of the number of results in the search neighborhood and S_r, where S_r represents the size of a neighborhood with neighborhood radius *r*. Assuming the size of the search neighborhood is *A*, the value of true match approximately equals to $\alpha A S_r$. Since each peer has one CN neighborhood signature, the value of target size in CN is the same as the size of the search neighborhood *A*. Therefore, *true_match/target_size* approximately equals to αS_r. Thus, the signature match ratio for CN is

$$\mu_{CN} = Fd + \alpha S_r(1-Fd).$$

For partial neighborhood superimposed (PN-S) signature scheme, each peer has *b* neighborhood signatures. Therefore, the value of target size in PN-S is *b* times of the size of the search neighborhood. Thus, signature match ratio for PN-S is

$$\mu_{PN-S} = Fd + \frac{\alpha S_r(1-Fd)}{b}.$$

In partial neighborhood appended (PN-A) signature scheme, the search signature is compared with each sub-signature in a neighborhood signature. Therefore, the value of target size in PN-A is S_r times of the size of the search neighborhood. The signature match ratio for PN-A is

$$\mu_{PN-A} = Fd + \alpha(1-Fd).$$

Based on [18], Fd can be obtained as

$$Fd = \left(1 - e^{\frac{-\omega s}{L}}\right)$$

where *w* represents the number of 1's set in a bit string, *s* represents the number of bit strings superimposed in a neighborhood signature, and *L* represents the length of a neighborhood signature. According to [19], the optimal false drop probability is achieved when

$$\omega =_{opt} = \ell n2 \cdot \frac{L}{s}.$$

The optimal false drop probability is approximately $0.5^{\omega_{opt}}$

4.1 Search

Next we derive the formulae for search message volume in both flooding and single-path search. The formulae for the three signature methods are derived, respectively. The differences and similarities among them are pointed out along the way.

4.1.1 Flooding Search

We first derive the number of search messages forwarded to nodes that are *i* hops away (denoted as Q_i), for each of the three signature methods. The search message is either extended within a neighborhood of a matched neighborhood signature or forwarded to the *(r + 1)*-hop-away peer nodes located right outside an unmatched neighborhood. In the following, we derive the number of search messages forwarded due to matched signatures first, then the ones due to unmatched signatures.

- **Calculating Q_i for CN**

  The number of search messages forwarded to nodes one hop away due to matched signatures is $b\mu_{CN}$. The derivation for the number of search messages forwarded to nodes that are more than 1 hop away is more complicated. The number of matched neighborhood signatures at hop distance $i-1$ is $Q_{i-1}(1-\beta)\mu_{CN}$, where $Q_{i-1}(1-\beta)$ is the number of visited nodes at hop distance $i-1$. For each matched neighborhood signature, the search message is forwarded to $b-1$ nodes (excluding the sender) at next hop distance to determine whether the match is a true match or not. The number of unmatched neighborhood signatures at hop distance $i-r-1$ is $Q_{i-r-1}(1-\beta)(1-\mu_{CN})$, where $Q_{i-r-1}(1-\beta)$ is the number of visited nodes at hop distance $i-r-1$. For each unmatched neighborhood signature, P_{r+1} search messages are extended to nodes $r+1$ hops away. Therefore, the complete formula for Q_i is

$$Q_i = \begin{cases} b\mu_{CN} & i=1 \\ Q_{i-1}(1-\beta)\mu_{CN}(b-1)+Q_{i-r-1}(1-\beta)(1-\mu_{CN})P_{r+1} & i>1 \end{cases} . \quad \textbf{(3)}$$

- **Calculating Q_i for PN-S**

  The number of matched neighborhood signatures at hop distance *i* (with *i > 1*) is $Q_{i-1}(1-\beta)(b-1)\mu_{PN-S}$, where $Q_{i-1}(1-\beta)$ is the number of visited nodes at hop distance $i-1$ and $b-1$ is the number of neighbor- hood signatures to be compared per node (excluding the one associated with the neighbor from which this peer got the search message). For each matched neighborhood signature, one search message is forwarded to the associated neighbor at next hop distance. Therefore the total number of search messages forwarded due to matched signatures is the same as the number of matched neighborhood signatures. The number of unmatched neighborhood signature at hop distance $i-r-1$ is $Q_{i-r-1}(1-\beta)b(1-\mu_{PN-S})$. Since a search message only needs to be forwarded to $(r+1)$-hop-away peers (i.e., those located right outside of the partial neighborhood corresponding to the unmatched neighborhood signature), the number of search messages forwarded to nodes $r+1$ hops away for each unmatched neighborhood signature is P_{r+1}. Therefore, Q_i for PN-S is

$$Q_i = \begin{cases} b\mu_{PN-S} & i=1 \\ Q_{i-1}(1-\beta)(b-1)\mu_{PN-S}+Q_{i-r-1}(1-\beta)(1-\mu_{PN-S})P_{r+1} & i>1 \end{cases} \quad \textbf{(4)}$$

- **Calculating Q_i for PN-A**

  The number of matched sub-signatures at hop distance *i* is $P_i\mu_{PN-A}$ and for each matched sub-signature, a search message is forwarded to the node associated to it. The number of unmatched signatures at hop distance $i-r-1$ is $Q_{i-r-1}(1-\beta)S_r(1-\mu_{PN-A})$ where $Q_{i-r-1}(1-\beta)$ is the number of visited node at hop distance $i-r-1$ and S_r is the number of signatures to be examined per node. Therefore, Q_i for PN-A is

$$Q_i = \begin{cases} b_{\mu_{PN-A}} & i=1 \\ \dfrac{P_{i\mu_{PN-A}}}{1-\beta} + \dfrac{Q_{i-r-1}(1-\beta)S_r(1-\mu_{PN-A})P_{r+1}}{b} & i>1 \end{cases} \quad \textbf{(5)}$$

Given Q_i for all three signature schemes, the search message volume can be expressed as

$$V_{search} = \sum_{i=1}^{T} M_s Q_i \,.$$

The message volume incurred for the response messages is:

$$V_{response} = \sum_{i=1}^{T}\left[M_r Q_i (1-\beta) + \sum_{j=i}^{T} s\alpha P_j \right].$$

The first term sums up the message volume incurred by the response message header and the second term sums up the message volume incurred by the result records. Since the response messages from the downstream neighbors are aggregated into one response message during the traversal back to the requester, the number of results arrived at one node is the total number of results returned from its downstream neighbors plus its own results, which denotes the meaning of $\sum_{j=i}^{T} \alpha P_j$.

4.1.2 Single-Path Search

Without signature files, the number of nodes that should process a search message in single-path search is $\min\left[\frac{R}{\alpha}, N\right]$ (min denotes minimum). Using signature files, the number of nodes that should be visited during the search process is $\frac{\min\left[\frac{R}{\alpha}, N\right]}{S_r}$. Taking into consideration of the redundant edges, the number of search messages is $\frac{\min\left[\frac{R}{\alpha}, N\right]}{S_r(1-\beta)}$. In addition, in order to determine whether a signature match is a true match or not, the peer nodes within the neighborhood need to further process the search message (i.e., the neighborhood checking process as described in Section 2.3.2). For each matched signatures, the neighborhood checking process incurs $\sum_{i=1}^{r} M_s Q$ message volume where Q_i is calculated according to the respective formula (i.e., the formulae (3)-(5)) of the three signature schemes for Q_i in flooding search. There are total $\frac{\mu \cdot \min\left[\frac{R}{\alpha}, N\right]}{S_r}$ matched signatures. Therefore, the formula for search message volume is

$$V_{search} = M_s \frac{\min\left[\frac{R}{\alpha}, N\right]}{S_r(1-\beta)} + \frac{\mu \cdot \min\left[\frac{R}{\alpha}, N\right]}{S_r} \sum_{j=i}^{r} M_s Q_i \,\cdot$$

Similarly, the response message volume is approximately

$$V_{response} = M_r \frac{\min\left[\frac{R}{\alpha}, N\right]}{S_r} + \sum_{i=1}^{\min\left[\frac{R}{\alpha}, N\right]} s\alpha S_r + \frac{\mu \cdot \min\left[\frac{R}{\alpha}, N\right]}{S_r} \sum_{i=1}^{r}\left[M_r Q_i (1-\beta) + \sum_{j=i}^{r} s\alpha P_j \right].$$

The first term and second term denote the message volume incurred by response message header and the result records in the initial search process, respectively. The last term is the message volume incurred by response messages generated during neighborhood checking process.

4.2 Peer Join

The join process for all three approaches is the same. When a node joins the networks, it first sends join messages to nodes in its neighborhood with radius *r*, incurring $\frac{S_r}{1-\beta}$ messages. Upon receiving join messages, the peer nodes in the neighborhood send their own local signatures to the newly joined node, incurring additional S_r messages. As explained in Section 2, when *r* is greater than 1 and the new peer joins the network through multiple connections, some peer nodes (within $r-1$ hops) that were far away from each other before the join are brought into the same neighborhood via the newly joined node. In this case, additional traffic is required to update the neighborhood signatures of other peer nodes within $r-1$ hops. Thus, the join message volume can be approximated as follow:

$$V_{join} = \begin{cases} M_j S_r \left(\frac{1}{1-\beta} + 1 \right) & \text{r = 1 or only one new connection} \\ M_j S_r \left(\frac{1}{1-\beta} + 1 \right) + M_j S_{r-1} \left(\frac{1}{1-\beta} + 1 \right) & \text{r > 1 and multiple new connection} \end{cases}$$

The formulae for join message volume of the three signature methods are the same as above. However, the sizes of join messages for CN, PN-S and PN-A are different. As can be observed from Figure3, in order for CN, PN-S and PN-A to have the same sized neighborhood signatures, the size of the sub-signatures that form PN-A neighborhood signatures is smaller than the size of sub-signatures for CN and PN-S, while the size of sub-signatures for PN-S is smaller than the size of sub-signatures for CN. Therefore, given the same storage size for all three signature schemes, the size of join messages for PN-A is smaller than the size for PN-S, which is smaller than the size for CN.

4.3 Peer Leave

When a node leaves the network, it first sends a leave message to nodes in its neighborhood with radius r, incurring $\frac{S_r}{1-\beta}$ leave messages. For CN, a node receiving the leave message sends a pseudo join messages to its neighborhood, which incurs $\frac{S_r}{1-\beta}$ messages with size as M_h. On receiving pseudo-join messages, peer nodes send back their local signature, incurring S_r messages with size as M_j . Therefore, the message volume incurred by each node to construct a new neighborhood signature in CN is $S_r\left(\frac{M_h}{1-\beta}+M_j\right)$the message volume for peer leave in CN is:

$$V_{leave}^{CN} = \frac{S_r M_l}{1-\beta} + S_r^2 \left(\frac{M_h}{1-\beta} + M_j \right). \tag{6}$$

Different from CN, when a peer leaves the network, only the peers on the affected branch need to construct a new neighborhood signature. Therefore, the message volume for peer leave in PN-S is:

$$V_{leave}^{PN-S} = \frac{S_r M_l}{1-\beta} + \frac{S_r^2}{b} \left(\frac{M_h}{1-\beta} + M_j \right). \tag{7}$$

In PN-A, when a node leaves, the affected nodes only need to delete the signature of the leaving node from its neighborhood signatures. Therefore, the message volume for peer leave in PN-A is

$$V_{leave}^{PN-A} = \frac{S_r M_l}{1-\beta}. \tag{8}$$

4.4 Peer Update

The formulae for the update message volume of the three signature schemes are similar to the formulae (6)-(8) for the leave message volume except that the M_l is replaced by M_u.

5 Performance Evaluation

In this section, we present the performance evaluation conducted to compare our proposal with some representative approaches for searching unstructured P2P networks. Total message volume is used as the primary performance metric in our simulation. We fist evaluate the performance of the proposed signature techniques analytically. In addition to analytical results, we also perform simulation based experiments. While the analytic results are obtained based on uniform network topology and uniform data distribution, the simulation experiments are conducted under both of uniform and power-law network topologies with both of uniform and non-uniform data distributions. Through these experiments, we examine the performance of neighborhood signature schemes, Gnutella, random walk, and local index[7] based on both of flooding and single-path searching strategies. The stop conditions for the flooding and single-path search are maximum search depth and minimum number of results, respectively. In the flooding experiments, we compare our signature mechanisms with Gnutella and local index, while in the single-path search we compare with random walk and local index. In addition, through these experiments we tested the sensitivity of the evaluated P2P searching approaches to various factors including *neighborhood radius, storage constraints, size of key attribute, number of data items at a node, search/update ratio, replication ratio, number of neighbors per peer* for uniform network topology, and *power-law network coefficient* for power-law network topology. In the following, we first describe the parameter setting for analytic experiments and simulation experiments. We then present the detailed experiment results.

5.1 Experiment Setup

As mentioned above, the analytic experiments are based on uniform network topology and uniform data distribution, in the simulation, we consider power-law network topology and nonuniform data distribution in addition to uniform network topology and uniform data distribution. Since the setups for analytic experiments and simulation experiments with uniform network topology and uniform data distribution are the same, we present them together and point out the settings that are unique to the simulations with power-law network topology and nonuniform data distribution when necessary in the following.

The simulations are initialized by generating signatures for 10000 pre-existing peers in the tested networks. A large number of operations consist of a randomized mix of search, peer join, peer leave, and peer update are injected into the P2P network in each experiment. The simulation experiments are run under both uniform and power-law network topologies. For the uniform network topology, the default average number of neighbors is set to 4, which is consistent with the average node degree in Gnutella [20]. We conduct experiments with average number of neighbors set to 8 as well. For the power-law network topology, based on measurement of Gnutella network [20], we set the power-law topology using a default coefficient of 1.4 if not specified otherwise. In one set of experiments, we also vary the power-law network coefficient to study the effect on the performance of various schemes. For both network topologies, we consider two different data distributions: uniform and nonuniform. Under the uniform data distribution, each node holds the same number of data items. We use the 80-20 rule to compose a nonuniform data distribution. That is, 80% of data items are distributed among 20% of the nodes, called *popular peers*, and the remaining 20% of data items are distributed among the remaining 80% nodes, called *unpopular peers*. Uniform data distribution is the default setting if not specified otherwise.

Table 3 lists the parameters and their default values used in the analytic experiments as well as simulation experiments. 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 [14]. Therefore we set the default 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. Since the size of data items itself is not a significant factor in differ-

[7]The implementation of local index follows the description in [19].

entiating the schemes under investigation, we use *size of key attribute* as an important parameter to characterize data items. 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 varying the number of data items per peer and 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) in order to observe their impacts on different search approaches. To simulate a generic data set in P2P systems, we use synthesized key attributes which are random integers (or composition of several random integers for the situations when the size of key attributes is larger than 4). The ratio of redundant edges is set to 30% according to [21] in the analytic experiments[8]. The default replication ratio and default search/update ratio are set at 0.5% and 10, respectively, while we also vary these two parameters in the experiments in order to study the effect on the performance of various schemes.

Table 3. Default parameter settings for the experiments

System parameters	
$N = 10000$	b = 4
$d = 400$	$k = 4$ bytes
$\alpha = 0.5\%$	$\beta = 30\%$
$\phi = 10$	
Stop conditions	
$T = 7$ (uniform), 5 (power-law)	$R = 1$
Message parameters (bytes)	
M_h = 80	M_j = 80 + index/signature size
M_S = *80 + length of search predicate*	M_r = *88 + s × number of results*
M_l = *84 (index/PN-A), 80 (CN/PN-S)*	M_u = *92 (index), 80 (CN/PN-S)* *84 + size of change record(PN-A)*

- **Stop condition settings:** The maximum search depth is set to 7 for uniform network topology. We ran some preliminary experiments and found out that in order to achieve the same search coverage in both uniform network topology and power-law network topology, the maximum search depth should be set to 5 in power law network. While we vary the replication ratio in one of the experiments to observe its impact on the single-path search, the minimum number of results is set to 1.
- **Message parameters settings:** The message header includes both the Gnutella message header and the TCP/IP header, which is 80 bytes [2]. 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). The search response message consists of the message header, the node identifier of the requesting peer, a counter indicating the total number of returned result pointers (a 4-byte integer) and a list of the result pointers. In addition to search messages and response 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 2.4. 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 2.4). To estimate the size of change record in analytic experiments, we make a conservative assumption that all the positions (represented by 4-byte integers) of the bits set in both old and new signatures are recorded in the change record. Thus, the size of the change record is estimated as $8W_{opt}$, where W_{opt} represents the optimal number of bits set in a signature file as explained in Section 4. 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, 2 bytes for the identifier of the data item to be updated, 2 bytes to indicate the position of the updated data attribute in the data item, and 4 bytes for the new key attribute value.

[8]This parameter is dependent on the specific network topology, i.e., the network size and number of neighbors per node, in the simulation. Thus, it is not preset separately.

5.2 Result

We conduct both analytic experiments and simulation experiments to examine the performance of proposed signature schemes. We found that the simulation results are consistent with the analytic results. In the following, we first briefly explain the analytic results and present one set of analytic results and simulation results side by side to demonstrate the consistency between them. Then for presentation clarity, we present detailed simulation results only.

5.2.1 Comparison Between Analytic Results and Simulation Results

We have the following observations based on the analytic model derived in previous section:

- With neighborhood radius increasing, the maintenance overhead incurred by neighborhood signatures increases while the search message volume either decreases or increases depending on the storage size.
- With storage size increasing, the search message volume incurred by neighborhood signatures improves while the signature maintenance overhead increases.
- The maintenance overhead incurred by PN-A is much smaller compared to the overheads incurred by PN-S and CN, which makes PN-A the best among the three proposed neighborhood signatures.

All of these observations are verified by simulation experiments as we will present later.

We find that our simulation results are consistent with the analytic results, which verifies the correctness of our simulation. In the following, we present one set of the analytic results and simulation results under both uniform and power-law network topologies when the storage size is set to 6.4KB and the neighborhood radius is varied from 1 to 5 in Fig. 4. We focus on the similarity between these results here while leaving the interpretation of these figures to next section. From this figure, we can see that the simulation results are very close to the analytic results. In addition, the simulation results under uniform network topology and power-law network topology are similar to each other.

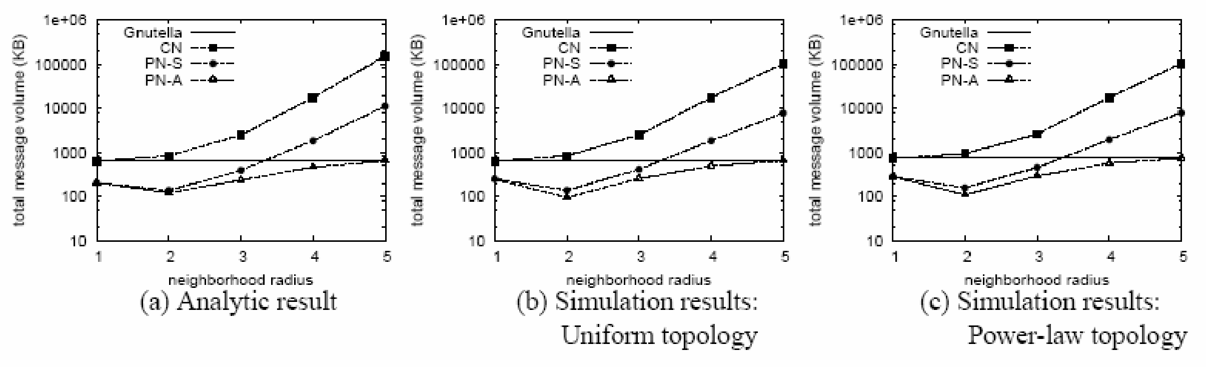


Fig. 4. Comparison of analytic results and simulation results under uniform and power law Network topologies. The y axis is on logarithmic scale for readability

In the following, the simulation results for flooding and single-path search are presented, respectively. For each of these two search strategies, we have conducted various simulation experiments under uniform and power law network topologies in combination of both uniform data distribution and nonuniform data distribution. The general trend observed from the results obtained under uniform topology is very similar to the one observed under power-law topology. Thus, we only present the results under power-law topology for presentation clarity.

5.2.2 Flooding Search

In the following, we first vary the neighborhood radius and storage size to compare the performance of the proposed signature schemes (Gnutella as a baseline) and to determine the best settings of those two parameters for those schemes. Then, we show the impacts of other parameters on the performance of Gnutella, local index, and our signature schemes. In the experiments, unless explicitly specified, the total message volume includes all message volume incurred by peer join, peer leave, peer update and search. We first present the results under uniform data distribution and then compare with the results under nonuniform data distribution.

Neighborhood Radius: In order to determine the best neighborhood radius for signature schemes under our experiment settings, we vary neighborhood radius from 1 to 5 and show the total message volume of CN, PN-S and PN-A with storage size 6.4KB in Fig. 5. We display the message volume for peer join, peer leave, peer update and search operations individually in Fig. 5 so that we can observe the effects of neighborhood radius on the search cost and signature maintenance cost. The y-axis is on a logarithmic scale for readability. The search message volumes incurred with neighborhood radius as 1 in signature schemes CN, PN-S, PN-A are 93%, 37%and 37% of the Gnutella traffic. When the neighborhood radius increases to 2, the search message volumes incurred by PN-S and PN-A decrease to 18% and 14%. This is due to the desirable (intended) filtering effect achieved by neighborhood signatures. When the neighborhood radius is increased further, more neighborhood information is forced to be compressed within the same storage space (either by superimposing more signatures in CN and PN-S, or appending more signatures with smaller size in PN-A), which increases the false positive probability. The end results is that the search message volume increases, 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 neighborhood radius values.

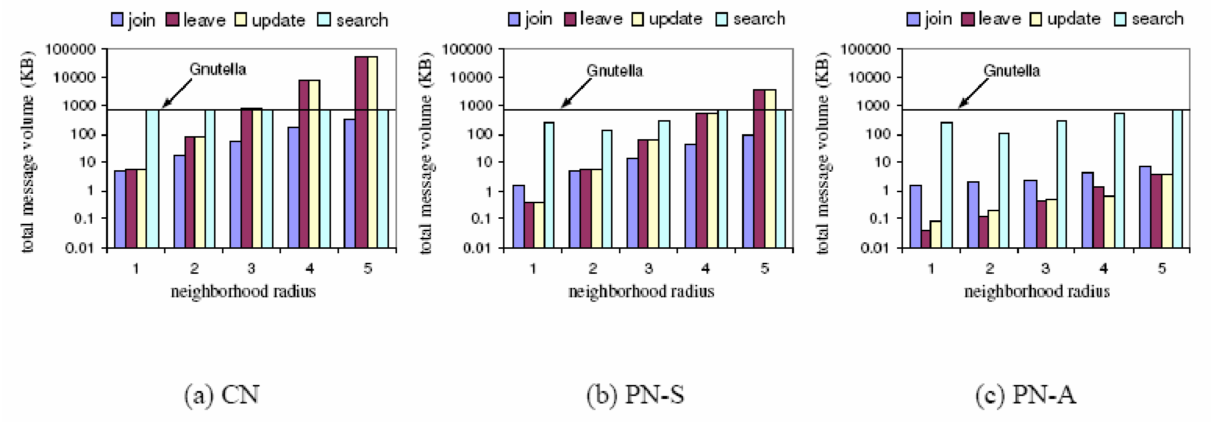


Fig. 5. Flooding: effect of neighborhood radius on signature schemes. The y axis is on logarithmic scale for readability

At the same 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 during peer leave/update. Between these two approaches, 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 compared to CN and PN-S.

Storage Size: Fig. 6 shows the total message volume as we allow various storage capabilities at each peer for the signatures. From this figure, we can see that when storage size increases, the total message volume decreases first, then increases again. The initial decrease is due to the better filtering effect of signatures with larger size. However, when the storage size keeps increasing, the maintenance overhead incurred by peer join/leave/update is overwhelming, resulting in increased total message volume. The minimum total message volume incurred by CN, PN-S and PN-A is 94%, 20% and 6% of Gnutella traffic when storage size is 1.6KB, 6.4KB, and 25.6KB, respectively.

The values shown in Fig. 6 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 storage size are given in Table 4). For CN, the optimal radius is 1 for all considered storage sizes as shown in Table 4. The reason is that when the signature size is small, join/leave/update cost is small and query cost dominates the total message volume as shown in Fig. 6(a). 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-S and PN-A, making a larger neighborhood radius more preferable in these two schemes when storage size is large, as shown in Table 4.

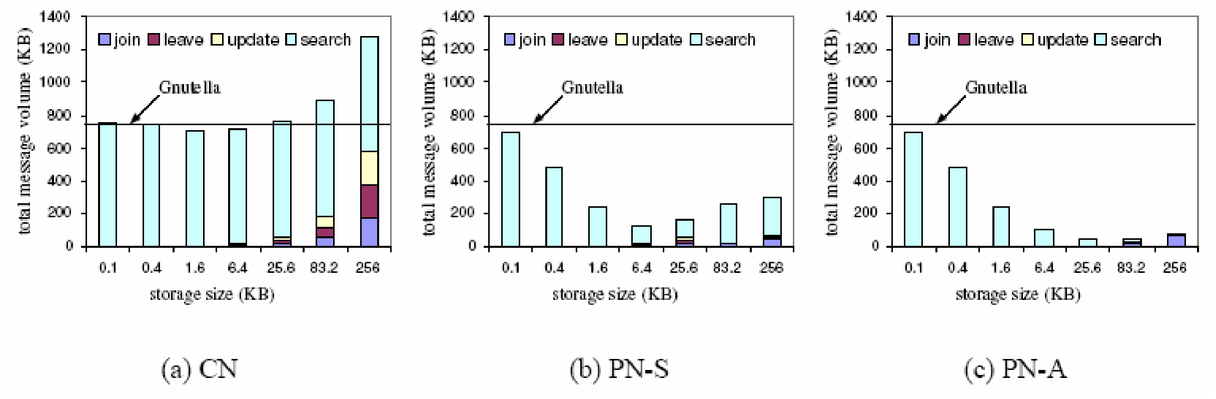


Fig. 6. Flooding: effect of storage size on signature schemes

Table 4. Flooding: optimal neighborhood radius for different storage sizes

Storage size(KB)	0.1	0.4	1.6	6.4	25.6	83.2	256
CN	1	1	1	1	1	1	1
PN-S	1	1	1	2	2	1	1
PN-A	1	1	1	2	3	4	4

Message Volume and Storage Tradeoff: Fig. 7 compares the performance of the three signature schemes with the local index, along with the Gnutella shown as a solid horizontal line, by increasing storage size. From this figure, we can observe that with storage size as small as 400 bytes, the partial neighborhood signature schemes (PN-S and PN-A) reduce message traffic by over 35% 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 with storage not less than 6.4KB. With a storage size of 6.4KB, the message volume of PN-A is only 40% of local index's traffic. As the storage space gets larger, index/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.

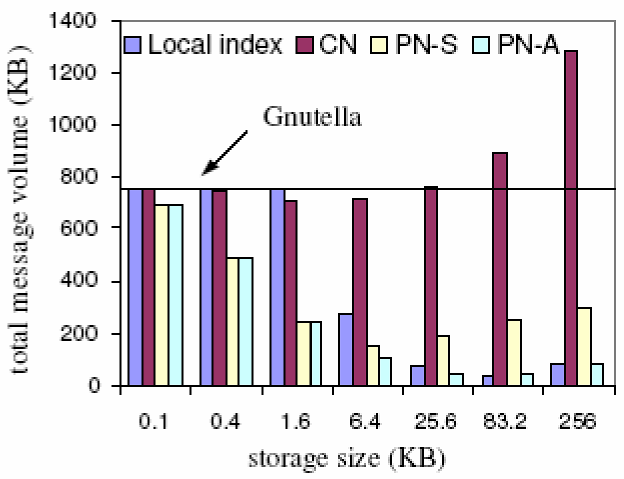


Fig. 7. Flooding: total message volume comparison among Gnutella flooding, local index, CN, PNS and PNA

Size of Key Attribute: Fig. 8 shows the total message volume with different sizes of key attribute. The y-axis is on a logarithmic scale for readability. In this simulation, we use increased attribute sizes to represent the situations where the (logical) key attribute consists of multiple keys or contains 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%[9]. It can be observed from the figure that the signature approaches outperform 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 18% 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 key attribute value(s). Local index performs well when the storage/total-attribute-size ratio is large. However, as the attribute size increases, the given storage size is not sufficient to index all data items[10]. Therefore, local index's performance is the same as Gnutella approach for larger attribute size.

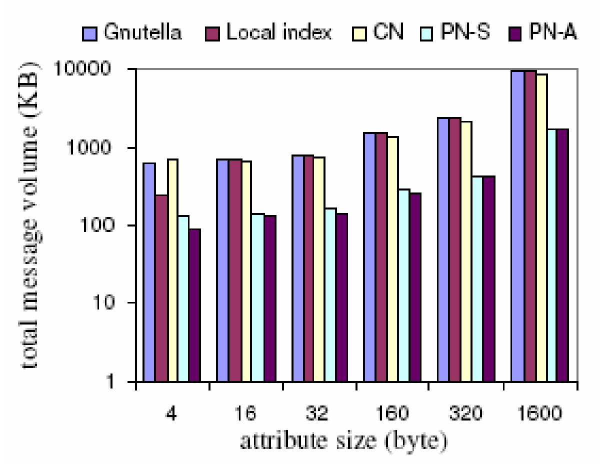


Fig. 8. Flooding: effect of key attribute size on Gnutella flooding, local index and signature approaches. The y axis is on logarithmic scale for readability

Number of Data Items: Fig. 9 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 Fig. 9 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 perform 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.

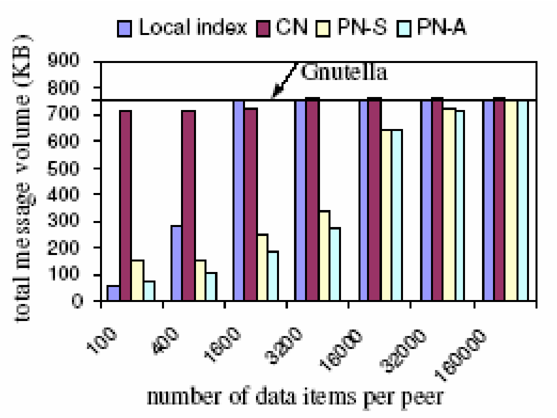


Fig. 9. Flooding: effect of number of data items per peer on Gnutella flooding, local index and signature approaches

[9]The storage/total-attribute-size ratio can be interpreted as the storage overhead normalized according to the total size of the key attributes of data items stored per peer.

[10]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 Fig. 8.

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: Fig. 10 shows the optimal total message volume of CN, PN-S and PN-A under different search/update ratios using two different update strategies (i.e., eager update and lazy update) as discussed in Section 2.4. The y axis is on logarithmic scale for readability. We can see from Fig. 10 (a) that when the eager update strategy is adopted, the total message volume decreases with increasing search/update ratio. This is due to the reduced signature maintenance overhead incurred by decreasing number of peer join/leave/update operations. The similar trend is observed when lazy update strategy is adopted as shown in Fig. 10(b). However, when the search/update ratio is smaller than 1 (i.e., there are more peer join/leave/update operations than search operations), the total message volume using lazy update strategy is much smaller than the one using eager update strategy, which demonstrates the benefits of lazy update in reducing the signature maintenance overheads when the peer join/leave/update operations are overwhelming.

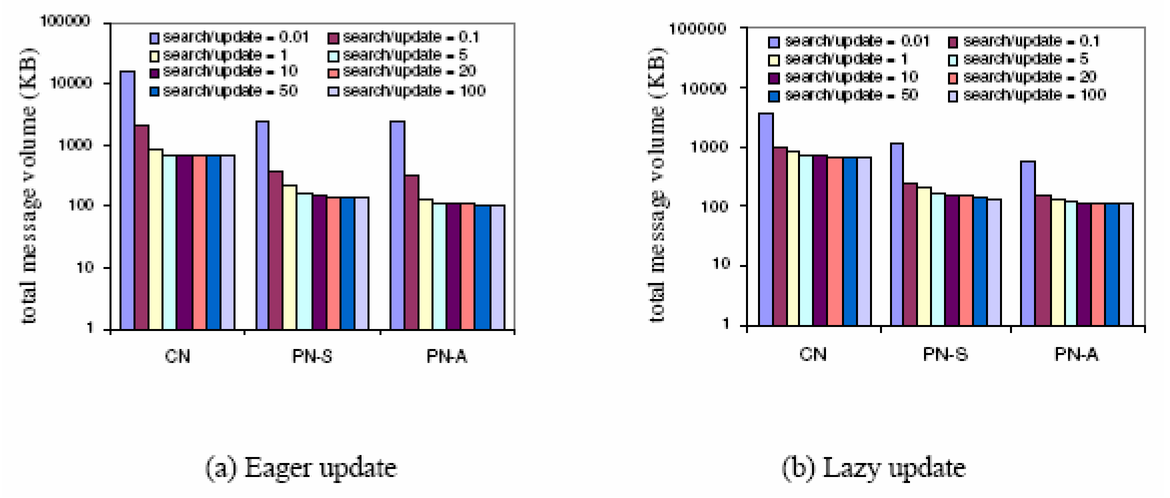


Fig. 10. Flooding: effect of search/update ratio on signature schemes. The y axis is on logarithmic scale for readability

Network Topology: Fig. 11(a) and Fig. 11(b) show the total message volumes of different approaches with storage size 6.4KB when the number of neighbors per peer is 4 and 8, respectively. From these two figures, we can see that the total message volume increases when the number of neighbors per peer increases from 4 to 8. In addition, with the number of neighbors per peer as 4, the optimal values for neighborhood radius in CN, PN-S and PN-A are 1, 2, and 2, while the optimal neighborhood radius for all neighborhood signatures becomes 1 with the number of neighbors per peer as 8. This is because with larger number of neighbors per peer, more information is compressed into the signatures with the same storage space, resulting in higher false positive probability, thus worse filtering effect.

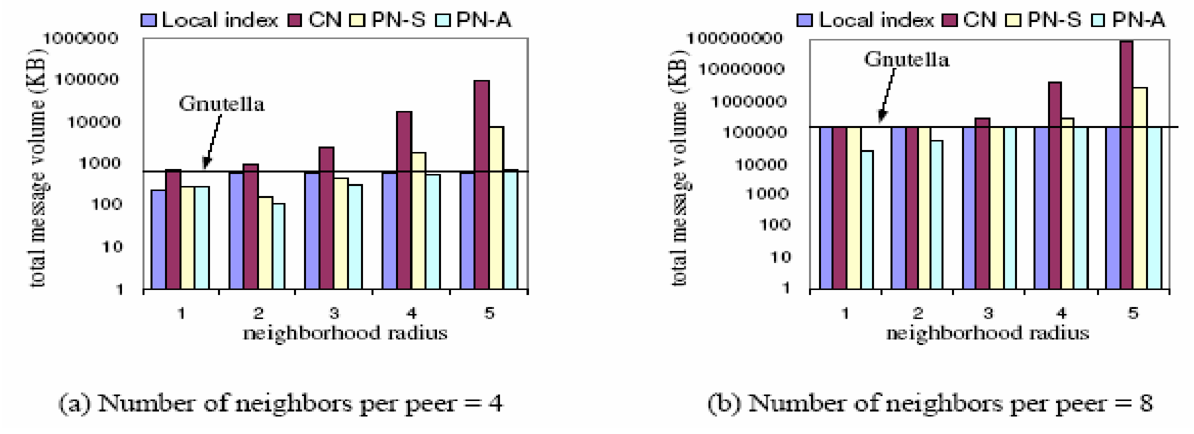


Fig. 11. Flooding: effect of number of neighbors per peer under uniform network topology on Gnutella flooding, local index and signature approaches

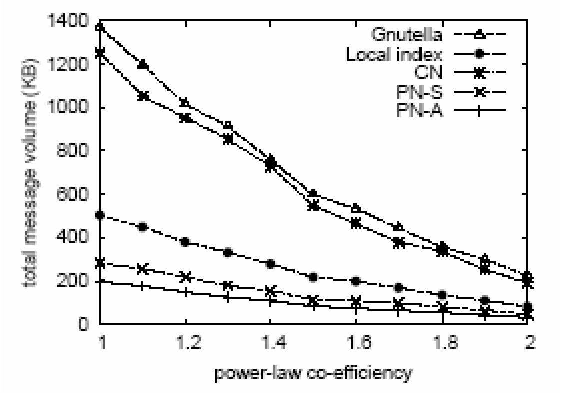


Fig. 12. Flooding: effect of power law network coefficient under power law network topology on Gnutella flooding, local index and signature approaches

Fig. 12 shows the effect of power-law network coefficient on the total message volume. From this figure, we can see that the total message volume decreases with the power-law network coefficient. The reason is that as power-law network coefficient increases, a few nodes have large number of neighbors, while majority of the nodes have a very small number of neighbors. Therefore, false positive probability of neighborhood signatures for majority of the nodes decreases, incurring reduced total message volume.

Data Distribution: Fig. 13 compares the performance of Gnutella flooding, local index and three signature schemes under uniform and nonuniform data distributions (as specified in Section 5.1). In this comparison, storage size for local index and signatures are set to be 6.4KB. For both Gnutella flooding and local index approach, there is no performance difference under these two different data distributions. For the signature schemas, the total message volume under nonuniform data distribution increases a little bit. This can be explained by the increased false positive probability of the neighborhood signatures which are contributed by *popular peers*. One important observation from Fig. 13 is that the performance of PN-A is better than local index under both uniform and nonuniform data distributions.

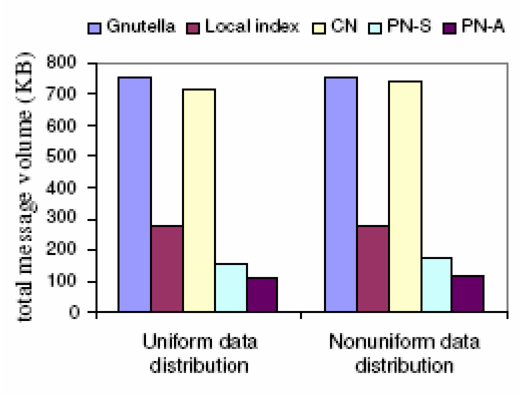


Fig. 13. Flooding: effect of data distributions on Gnutella flooding, local index and signature approaches

5.2.3 Single-Path Search

In addition to the seven parameters (neighborhood radius, storage size, key attribute size, number of data items, search/update ratio, number of neighbors per peer, and power-law network coefficient) 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. We compare the performance of the proposed signature schemes with random walk and local index. The general trend observed from the results is similar to that observed for flooding. For presentation clarity, we only present the comparison among random walk, local index and signature schemas when storage size and replication ratio increase, respectively.

Message Volume and Storage Tradeoff: Fig. 14 shows the total message volume comparison among random walk, local index, CN, PN-S and PN-A. Once again, we find that 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.

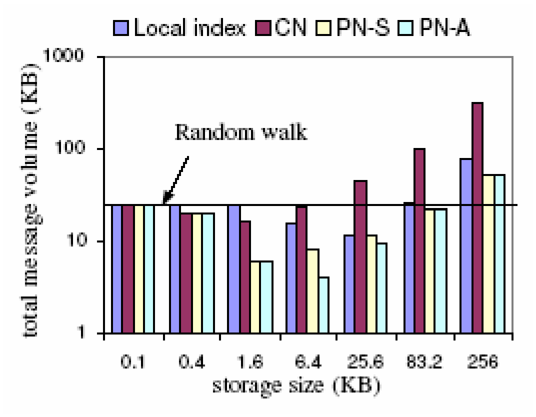


Fig. 14. Single path: total message volume comparison among random walk, local index, CN, PNS and PNA. The y axis is on logarithmic scale for readability

Replication Ratio: Fig. 15 compares random walk, local index and PN-A for different degrees of replication of data items in the network. The y-axis is on a logarithmic scale for readability. 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, as expected, random walk can perform rather well, since there is a higher likelihood of finding the requested 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 local index approaches which can direct searches in a more productive manner. Of these two approaches, 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 under a replication ratio of 0.1% and only 17% of random walk traffic under a replication ratio of 0.5%. Compared to local index, PN-A incurs 29% of local index traffic under a replication ratio 0.1% and 43% of local index traffic under a replication ratio 0.5%.

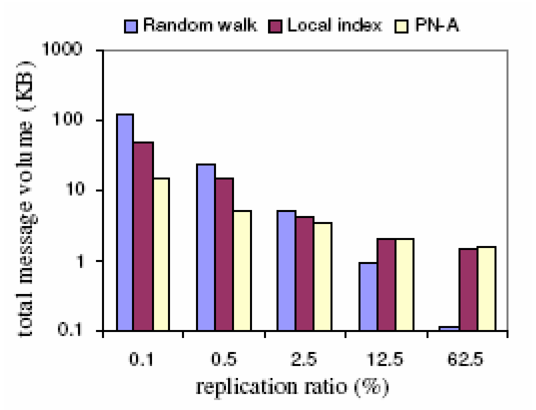


Fig. 15. Single path: effect of replication ratio on random walk, local index and signature approaches. The y axis is on logarithmic scale for readability

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 generate when searching for resources/information. One can argue that with infinite storage capacity it is possible to maintain complete auxiliary information for 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 for updating the auxiliary information when nodes join/leave the system or when data content is updated at a peer node, which can happen quite frequently in a dynamic P2P system. This trade-off between storage space vs. network traffic opens up a rich research space to explore. Previous research has looked into one possible mechanism, local index, in this space. In this paper, we propose three new mechanisms based on signature files within this space that can provide a better focused search at a lower storage overhead than local index. We have shown that, with a very small storage cost, signatures are quite effective at reducing search costs compared to local index. In addition, the message overheads of join/leave/update operations are adequately compensated by the savings in search messages. Of the three schemes, CN, PN-S and PN-A, that we propose, PN-A gives the best performance.

The schemes have been extensively evaluated through analytical experiments and simulation experiments both with the intention of fine-tuning the parameters that they use (neighborhood radius, storage size) and comparing with the previous proposals using both flooding and single-path search strategies under different network topologies (uniform and power-law), different sizes of key attribute, different number of data items at a peer, different data distribution patterns, different degrees of data replication, and different proportions of operations (search/update). We uniformly find PN-A gives 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) Signature approaches can search across multiple attributes by appropriately encoding all the attributes when composing a signature, instead of being restricted to one or a small number of attributes which needs to be predetermined as in index approach. This facilitates keyword and content based search. (b) It takes a certain minimum amount (threshold) of storage to store an index. With storage size less than this threshold, index approach can not be used 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). 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. While we have demonstrated in [13] that semantic clustering can improve performance significantly in structured P2P overlays, we are also investigating semantic clustering via signature approaches in unstructured overlays. Finally, we are investigating P2P applications overlaid on wireless networks.

References

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

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

[3] D. Kossmann, “The state of the art in distributed query processing,” *ACM Computer Survey*, Vol.32, No.4, pp.422–469, 2000.

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

[5] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, “Search and replication in unstructured peer-to-peer networks,” *Proceedings of ACM International Conference on Supercomputing*, pp. 84–95, June 2002.

[6] S. Ratnasamy, P. Francis, M. Handley, R. M. Karp, and S. Schenker, “A scalable content-addressable network,” *Proceedings of ACM SIGCOMM*, pp. 161–172, August 2001.

[7] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, “Chord: A scalable peer-to-peer lookup service for Internet applications,” *Proceedings of ACM SIGCOMM*, pp. 149–160, August 2001.

[8] A. I. T. Rowstron and P. Druschel, “Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems,” *Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pp. 329–350, November 2001.

[9] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz, "Tapestry: A Resilient Global-scale Overlay for Service Deployment," *IEEE Journal on Selected Areas in Communications,* Vol. 22, No. 1, pp. 41-53, January 2004.

[10] M. Li, W.-C. Lee, and A. Sivasubramaniam, "Semantic Small World: An overlay network for peer-to-peer search," *Proceedings of International Conference on Network Protocols (ICNP)*, pp. 228–238, October 2004.

[11] Morpheus website. http://www.musiccity.com.

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

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

[14] B. Yang and H. Garcia-Molina, "Improving search in peer-to-peer networks," *Proceedings of International Conference on Distributed Computing Systems (ICDCS)*, pp. 5–14, July 2002.

[15] M. Li, W.-C. Lee, and A. Sivasubramaniam, "Neighborhood signatures for searching P2P networks," *Proceedings of International Database Engineering and Application Symposium (IDEAS)*, pp. 149–158, July 2003.

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

[17] F. M. Cuenca-Acuna, C. Peery, R. P. Martin, and T. D. Nguyen, "PlanetP: using gossiping and random replication to support reliable peer-to-peer content search and retrieval," *Proceedings of the 12th International Symposium on High Performance Distributed Computing (HPDC)*, pp. 236–249, June 2003.

[18] C. Faloutsos and S. Christodoulakis, "Signature files: An access method for documents and its analytical performance evaluation," *ACM Transaction on Office Information Systems*, Vol.2, No.4, pp.267–288, October 1984.

[19] S. Staissny, "Mathematical analysis of various superimposed coding methods," *American Documentation*, Vol.11, No.2, pp.155– 169, 1960.

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

[21] B. Yang and H. Garcia-Molina, "Designing a super-peer network," *Proceedings of International Conference on Data Engineering (ICDE)*, pp.49–62, March 2003.