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 to use signatures for directing searches along selected network paths, 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. Extensive 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 much better performance at a small storage cost.*

1 Introduction

The advent of facilities such as Napster [3] and Gnutella [1] 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, these Peer-to-Peer (P2P) systems can employ the host nodes to themselves acting as servers for other nodes. Despite avoiding centralized server bottlenecks and single points of failure, the P2P systems present interesting challenges in locating data items among these numerous host nodes. The centralized server in Napster [3], 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 among the nodes and/or exploit the topology of the P2P overlay network to perform certain kind of search ordering which can help in getting to the requested data items. CAN [11], Chord [13], Pastry [12], Tapestry [16] and P-Grid [4, 5] are examples of systems using such a strategy to control the number of hops that need to be traversed to get to the requested 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 the above techniques less suitable for these unstructured and dynamic environments, and are consequently not under consideration here.

There are two main strategies that have been proposed/explored for searching in decentralized and dynamic environments of P2P systems without relying on the network topology and data placement:

- **Strategy 1:** This strategy lets messages poll nodes, without having any idea of where the data may be held by the destination nodes, till the required items are found. Gnutella [1] and *random walk* [6, 10] use such a strategy. The down side of this strategy is the possible network overload due to a large number of generated search messages (Gnutella) or a long latency to satisfy a request (random walk).

- **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 additional information is typically index over the data that are contained either within hierarchical clusters [2] or by nearby neighbors [7, 9, 14]. This indexing approach requires determining what attributes to index a priori and thus constraining the allowable search, in addition to the high space cost that is incurred in storing the index itself.

Proceedings of the Seventh International Database Engineering and Applications Symposium (IDEAS'03)
1098-8068/03 $17.00 © 2003 **IEEE**

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 storage 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 storage 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 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. Naturally, the signature technique is very suitable for filtering information stored in 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. Their merits in reducing network traffic are extensively evaluated for content search, node join, leave and update operations. These schemes are compared with the current state-of-the-art approaches, Gnutella [1] and Random Walk [6, 10] for strategy 1, and Local Index [14] for strategy 2. Our simulations show that the signature approaches are much better than these alternatives for most reasonable storage space availability assumptions on host nodes.

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. Section 4 gives the experimental setup for the evaluation and Section 5 details results from experiments under 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. 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, namely *peer update*.

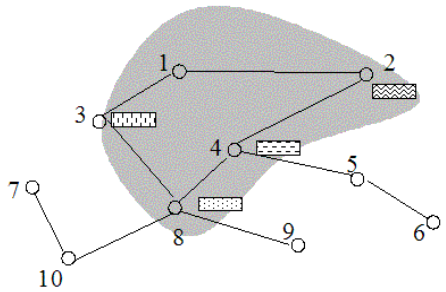


Figure 1. A partial snapshot of a P2P network.

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 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. 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 *neighborhood*, which includes all the peers reachable within a given distance. 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.

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 net-

[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.

Proceedings of the Seventh International Database Engineering and Applications Symposium (IDEAS'03)
1098-8068/03 $17.00 © 2003 **IEEE**

work. 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 are typically used to stop excessive spreading of a search:

- **Maximum search depth**: This stop condition is used in Gnutella [1]. A preset time-to-live value (TTL) is included in the search message to keep track of maximum remaining search depth. 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 [6, 10]. 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 Metrics

Since the primary issue investigated in this paper is the trading-off of storage space for reducing network traffic, we use the following metric to evaluate various P2P search techniques discussed in this paper:

- **Total message volume:** For signature schemes (as well as index schemes), besides 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 total number of messages and the size of different messages (including search messages and messages incurred for signature maintenance during peer join/leave/update), as the performance metric.

3. Neighborhood Signatures

In this section, we first provide some preliminary 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 content offered within the neighborhood of a peer. This helps direct the search to a subset of the nodes, which are probabilistically more productive, while not requiring as much storage as index approaches. We 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).

3.1 Preliminaries

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 superimposing them together[2]. Figure 2 depicts the signature generation and comparison processes of a digital file and some searches.

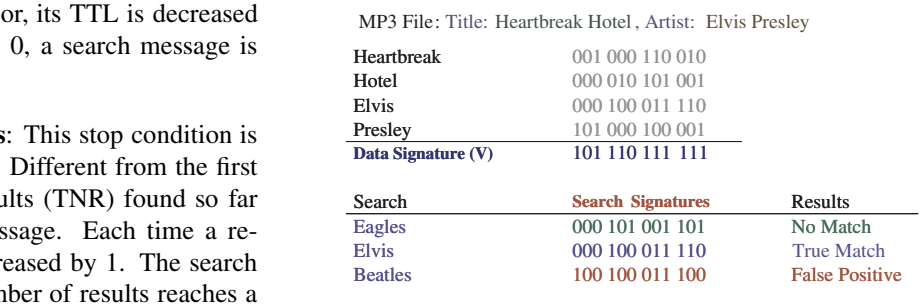


Figure 2. Illustration of Signature Generation and Comparisons.

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, 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*. The space devoted to the signature can influence the probability of false positives. Obviously the matched documents still need to be checked against the search criteria to distinguish a true match from a false positive.

3.2. Proposed Signature Schemes for P2P System

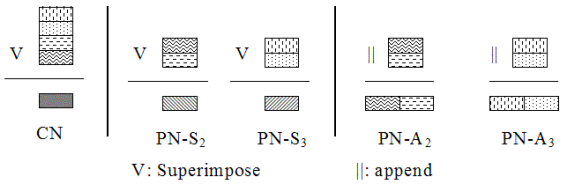
Before proceeding to introduce the proposed signature schemes, we first assume that a *local signature* is created at each peer of a P2P network to index the local content available at the peer. By doing this, search over the local content of a peer is processed efficiently. Furthermore, A peer

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

Proceedings of the Seventh International Database Engineering and Applications Symposium (IDEAS'03)
1098-8068/03 $17.00 © 2003 **IEEE**

may collect and maintain auxiliary information regarding digital content available within a specific network distance (i.e., its neighborhood). Therefore, a peer 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. Figure 1 shows a partial snapshot of a P2P network with the local signatures of peer 2, 3, 4, and 8 represented by rectangles with different filling patterns. Figure 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. 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

Figure 3. Illustration of neighborhood signature generation: (a) CN; (b) PN-S; (c) 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 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 *PN-S signature*. Figure 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 peer 2 and 4 and PN-S_3, the neighborhood signature for branch 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 a clue of which peers should be visited, resulting in searching all of these peers. An alternative that we propose, called *PN-A Signature*, is to append (concatenate) all of the local signatures within a branch of the neighborhood into a partial neighborhood signature[3]. 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. Figure 3(c) shows that peer 1 has 2 PN-A signatures, where 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.

3.3. Search Algorithms

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 visits randomly chosen nodes 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.

3.3.1 Flooding Search

In this section, we describe how a peer utilizes neighborhood signatures to perform searches based on 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

[3] 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.

Proceedings of the Seventh International Database Engineering and Applications Symposium (IDEAS'03)
1098-8068/03 $17.00 © 2003 **IEEE**

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: 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 search message.
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
```

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 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 x hops away (the neighborhood) need not be checked. As a result, the search message is dropped when $TTL \leq x$. In this case, 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.

The flooding search algorithms for the two partial neighborhood (PN) signature schemes are only slightly different from the one discussed above (refer to line 14-19). 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 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 $x + 1$ hops away, located right outside of the partial neighborhood corresponding to this neighborhood signature.

3.3.2 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). Due to space constraints, we omit the detailed algorithm here. 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*). 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 matched neighborhood signatures in PN-S, or directly to the peers with matched local signatures in PN-A.

3.4 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 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.

Proceedings of the Seventh International Database Engineering and Applications Symposium (IDEAS'03)
1098-8068/03 $17.00 © 2003 **IEEE**

- **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 its 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. 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 affected peers only need to update the relative subsignatures in their neighborhood signatures.

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. We consider two different network topologies, *uniform* and *power-law*, which have been studied in related work as well [10, 14]. Based on [8], we set the power-law topology using an exponential efficiency of 1.4. For both network environments, we consider two different data distribution patterns, uniform and nonuniform data distribution. Under uniform data distribution, each node holds the same number of data items. Under nonuniform data distribution, 80% data items are distributed among 20% of the nodes, called as *popular peers*, and the remaining 20% data are distributed among the remaining 80% nodes, called as *unpopular peers*. 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 operations to the other operations that require 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 total number of peers in the network.

Now we present the parameters and their values used in the simulations with the justification for these choices.

- **System parameter settings:** The average number of shared files per peer has been observed to be around 340 in [14], 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. Since the size of data items itself is not a significant factor in differentiating 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 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. To simulate a generic environment in P2P systems, we use synthesized key attributes generated by a random number generator in the simulation. The average number of neighbors is set to 4, which is consistent with the average node degree in Gnutella [15]. 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

Proceedings of the Seventh International Database Engineering and Applications Symposium (IDEAS'03)
1098-8068/03 $17.00 © 2003 **IEEE**

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 [8]. 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.

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 power-law network topology under uniform data distribution as well as nonuniform data distribution. We have also conducted simulation based on uniform network topology. The general trend observed from the results with uniform topology are similar to the one with power-law topology, so we omit it due to space constrains.

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. Total message volume, as discussed in Section 2.2, is used as the primary performance metric in our simulation.

5.1. Flooding

In the following experiments, we first vary the neighborhood radius and storage size to compare the performance of the proposed signature methods and to determine the best settings of those two parameters for the signature schemes. Then, we show the impacts of the other parameters as mentioned in Section 4 on performance of Gnutella, local index, and our signature methods. In the experiments, unless explicitly specified, search/update ratio is set at 10. We first present the results under uniform data distribution and then compare with the results under nonuniform data distribution.

Optimal neighborhood radius: Table 1 shows the values of neighborhood radius that give the lowest message volume for the given storage size (referred to as optimal radius). For CN and PN-S, the optimal radius is 1 for all considered sizes as shown in Table 1. The reason is that when the signature size is small, join/leave/update cost is small and query cost dominates the total message volume (figures are omitted due to space constraints). 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 larger neighborhood radius more preferable in this scheme when storage size is large, as shown in Table 1.

Table 1. Flooding: optimal neighborhood radius that generates the minimum total message volume for different storage sizes.

Storage size	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

Size of Key Attribute: Figure 4 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%[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 attribute size 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.

Number of Data Items: Figure 5 shows the total message volume as we allow the number of data items

[4]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.

[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 4.

Proceedings of the Seventh International Database Engineering and Applications Symposium (IDEAS'03)
1098-8068/03 $17.00 © 2003 **IEEE**

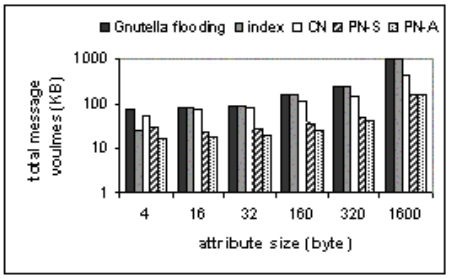


Figure 4. Flooding: effect of key attribute size on signature approaches. The y-axis is on logarithmic scale for readability.

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 5 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.

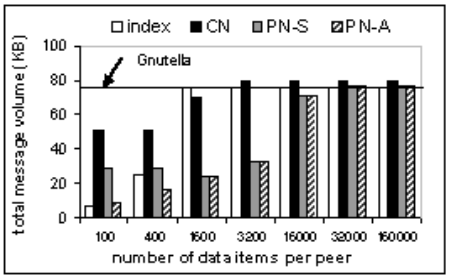


Figure 5. Flooding: effect of number of data items per peer on signature approaches. The total message volume of Gnutella is shown by the horizontal line.

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 6 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.

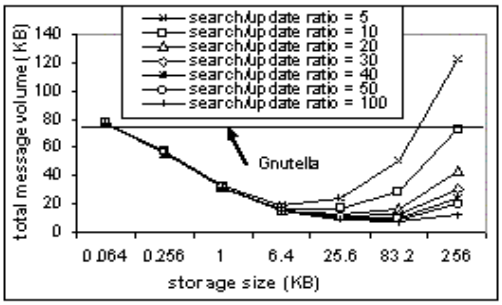


Figure 6. Flooding: effect of search/update ratio on signature approaches. Total message volume of Gnutella is shown by the horizontal line.

Message Volume and Storage Tradeoff: Figure 7 compares the performance of the three signature schemes with the index based approach, along with the Gnutella shown as a solid horizontal line, by increasing storage size. From the figure, 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%, 45% and 42% of the Gnutella traffic. With a storage size of 6.4KB, all three signature schemas 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/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.

Data Distribution: Figure 8 compares the performance of Gnutella flooding, local index and three signature schemas under uniform and nonuniform data distributions (as specified in Section 4). In this comparison, storage size for index and signatures are set to be 6.4KB. For both Gnutella flooding and index approach, there is no performance difference under these two different data distributions. For the signature schemas, the total message volume

Proceedings of the Seventh International Database Engineering and Applications Symposium (IDEAS'03)
1098-8068/03 $17.00 © 2003 **IEEE**

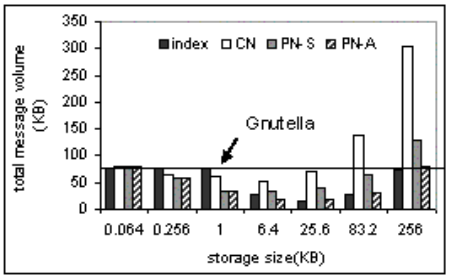


Figure 7. Flooding: total message volume comparison among local index, CN, PN-S and PN-A. Total message volume of Gnutella is shown by the horizontal line.

under nonuniform data distribution is increased 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 Figure 8 is that the performance of PN-A is better than local index under both uniform and nonuniform data distributions.

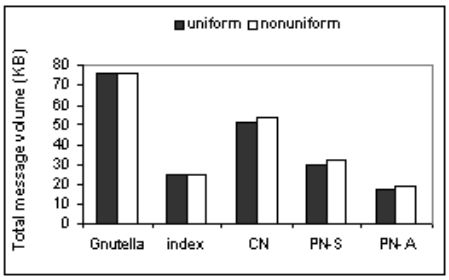


Figure 8. Flooding: effect of data distributions on Gnutella flooding, index and signature approaches.

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. The results are similar to that observed for flooding. Due to space constrains, we only present the comparison among random walk, index and signature schemas when the storage size and replication ratio increases, respectively.

Message Volume and Storage Tradeoff: Figure 9 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.

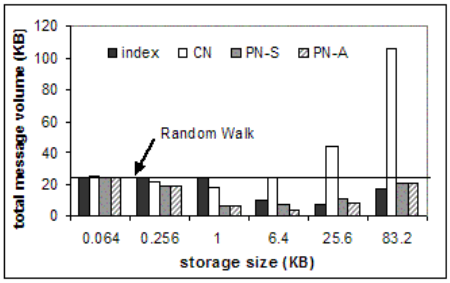


Figure 9. Single-path: total message volume comparison among local index, CN, PN-S and PN-A. Total message volume of random walk is shown by the horizontal line.

Replication Ratio: Figure 10 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 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 index traffic under a replication ratio 0.1% and 43% of index traffic under a replication ratio 0.5%.

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. We have proposed three new mechanisms based on signature files to facilitates search in p2p systems.

Proceedings of the Seventh International Database Engineering and Applications Symposium (IDEAS'03)
1098-8068/03 $17.00 © 2003 **IEEE**

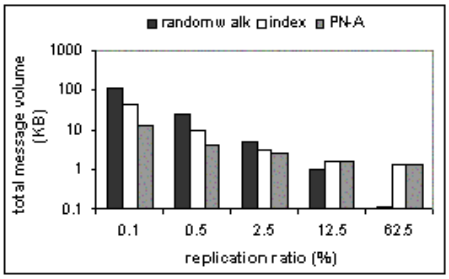


Figure 10. Single-path: effect of replication ratio on random walk, local index and signature approaches. The y-axis is on logarithmic scale for readability.

The schemes have been extensively evaluated and 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) Signature approaches 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 as in index approach. This facilitates keyword and content based search, etc. (b) It takes a certain minimum amount (threshold) of storage to compute (and 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, 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 peer join/leave/update 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] Gnutella website. http://gnutella.wego.com.

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

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

[4] 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.

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

[6] 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.

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

[8] 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.

[9] 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, November 2000.

[10] 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.

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

[12] 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.

[13] 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.

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

[15] B. Yang and H. Garcia-Molina. Designing a super-peer network. In *Proceedings of the 19th International Conference on Data Engineering (ICDE)*, March 2003.

[16] 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.

Proceedings of the Seventh International Database Engineering and Applications Symposium (IDEAS'03)
1098-8068/03 $17.00 © 2003 **IEEE**