SemPGRID'04

A Small World Overlay Network for Semantic Based Search in P2P Systems

Mei Li[1], Wang-Chien Lee[1], Anand Sivasubramaniam[1], and Dik Lun Lee[2]

[1] Pennsylvania State University, University Park, USA
{meli, wlee, anand}@cse.psu.edu
[2] Hong Kong University of Science and Technology, Hong Kong
dlee@cs.ust.hk

Abstract. For a peer-to-peer (P2P) system holding massive amount of data, efficient search for resources (such as data or services) is a key determinant to its scalability. This paper presents *semantic small world (SSW)*, an overlay network and index structure for semantic based P2P search. By dynamically clustering peer nodes in a semantic space based on the semantics of their data and organizing the clusters into a small world network, SSW achieves a very competitive trade-off between the search latencies/traffic and maintenance overheads. Preliminary evaluation shows that SSW is much more scalable to very large network sizes and very large numbers of data objects compared to pSearch, the state-of-the-art semantic-based search technique for P2P systems.

1 Introduction

The advent of applications such as Napster and Gnutella has made peer-to-peer (P2P) systems popular for the wide-spread exchange of resources and voluminous information between thousands of users. In contrast to traditional client-server computing models, a node in these P2P systems can act as both a server as well as a client. Despite avoiding centralized server bottlenecks and single point of failure, these decentralized systems present fundamental challenges when searching for resources (e.g. data, files, and services) available at one or more of these numerous host nodes. Meanwhile, such decentralization mandates that these systems dynamically adapt to continuous node membership and content changes without incurring high maintenance overheads.

The importance of P2P searches has motivated several proposals for performing these operations efficiently. Mechanisms such as Gnutella and Random Walk [8] either flood the network or search through a single path in the network randomly. While their search costs may not be low in terms of the total number of messages and/or the number of hops traversed per search, their advantages are in the low maintenance cost, making it relatively easy to handle membership changes[3], or even data content changes. Improvements to better direct such

[3] In this paper, peer join, peer leave and peer failure are collectively referred to as membership changes.

messages by indexing around a neighborhood (of an overlay network), such as Local Index [16] and Neighborhood Signature [7], can enhance the performance of searches. However, membership/content changes can require additional costs to update such indexes. Further improvements to search efficiency has led to constructing overlay networks (e.g., CAN [12], CHORD [14], TAPESTRY [13], PASTRY [18]) that *use hashed keys on the data name space* to direct the searches to the specific node(s) holding the requested data objects. This comes at a higher maintenance cost for updating the relevant information on membership/content changes.

While all these techniques address the scalability issue (particularly of searches) with respect to the number of nodes in the P2P system, it is equally important to address the voluminous information content of such systems. The vast repositories of information (just as in the Internet today) are simply not favorable to searches based on hashed keys, mandating the employment of *content/semantic based searches*. This is the reason why search engines are popular for navigating the Internet today.

Recognizing the need for content-based search in P2P networks, a recent proposal called *pSearch* [15] builds a semantic-based information retrieval engine on top of an m-dimensional CAN overlay network. To address the issues of high dimensionality, pSearch stores the physical location of each data object in p places in the CAN based on p separate partial semantic spaces. This structure incurs not only a considerable amount of state information (each node requires maintenance of $2m$ pointers as mandated by CAN) but also a nontrivial cost for publishing index information to the CAN (each data object needs to be published to the CAN p times when newly created by an existing node or brought in by an incoming node). Moreover, a search requires all p separated spaces to be examined.

To address many of these problems, this paper presents a novel approach, *Semantic Small World (SSW)*, to build a P2P system from the ground-up. The idea is to dynamically *cluster* peers with semantically similar data closer to each other in a semantic space and organize the clusters into an overlay network and a distributed index structure in the process of peer joins and leaves[4]. We show a simple yet effective dimensionality reduction technique (called *adaptive space linearization*) for constructing a one dimensional SSW to address the challenges raised by high dimensionality of semantic space. The overlay network that we construct is a *small world* network that has a much more attractive trade-off between search path length and maintenance costs compared to other overlays such as CAN and CHORD. In addition, SSW is an index structure that supports all dimensions of the semantic space in the search instead of limiting itself to perform p separate searches on m-dimensional CAN overlay network (as is done in pSearch). This approach can potentially lead us to data objects that may be missed out in pSearch. Preliminary evaluation shows that SSW is much more scalable to very large network sizes and very large numbers of data objects

[4] Distributed index and overlay network are two synonymous facets of the SSW, so we use them interchangeably.

compared to pSearch, the state-of-the-art semantic-based search technique for P2P systems.

The rest of this paper is structured as follows. Background on semantic space, semantic vector, small world network and related work are provided in Section 2. In Section 3, the concept of SSW and technical challenges are presented. Algorithms for various P2P operations are detailed in Section 4. The simulation setup and results for a preliminary performance evaluation are presented in Section 5 and Section 6, respectively. Finally, we conclude this paper and outline directions for future research in Section 7.

2 Preliminaries

In this section, we provide background on semantic space/vector and small world network and review some studies related to our work.

2.1 Background

Semantic Space and Vector. Various digital objects, such as documents, multimedia, and genomic data can be represented and stored as data objects in P2P systems. The semantics or features of such data object can be identified by a k-element vector, namely *Semantic Vector (SV)* (also called as *feature vector* in the literature)[5]. Each element in the vector represents a particular feature or attribute associated with the data object (i.e., color for an image, concept or key word for a text document) with weight representing the importance of this feature element in representing the semantics of the data object. SV can be obtained or derived from attributes, content or metadata of the object and the typical approaches to generate SV includes completely representing the data object, extracting some properties of the object, or deriving some statistics of the object. The SV of a data object can be mapped to a point in a k-dimensional *semantic space*. Euclidean distance is used to represent the *semantic distance* between two SVs in this paper.

Small World Network. Small world networks can be characterized by *average path length* between two nodes in the network and *cluster coefficient* defined as the probability that two neighbors of a node are neighbors themselves. A network is said to be small world if it has small average path length (i.e., similar to the average path length in random networks) and large cluster coefficient (i.e., much greater than that of random networks). Studies on a spectrum of networks with small world characteristics show that searches can be efficiently conducted when the network exhibits the following properties: 1) each node in the network knows its local neighbors, called *short range contacts*; 2) each node knows a small number of randomly chosen distant nodes, called *long range contacts*, with

[5] Here the data objects are not limited to specific types. Therefore, we use generic semantic vectors, instead of specialized ontology developed for certain type of data objects (such as semantic web), to represent the semantics of data objects.

Table 1. Comparison of P2P Search Techniques.

Schemes	Gnutella	Local Index	CAN	CHORD	Semantic Small World
Search cost (total messages)	O(N)	O(N/h)	O($kN^{1/k}$)	O($logN$)	O($\frac{1}{l}log^2(N/M)^{1/k} + M$)
States maintained per peer	0	O(h)	O($2k$)	O($logN$)	$O(2k + l + M)$

probability proportional to $\frac{C}{d}$ where d is the distance and C is a normalization constant that brings the total probability to 1 [5][6]. A search can be performed in $O(log^2N)$ steps on such networks, where N is the number of nodes in a network [6]. The constant number of contacts (implying low maintenance cost) and small average path length serve as the motivation for trying to build a small world overlay network in our approach.

In addition to our proposal of constructing a small world network, [4] mentioned the idea of using small world principles to facilitate search in P2P without giving rigid treatment on how to construct a small world and how search can be conducted. [9] proposes to build a one-dimensional small world network based on the principles discussed above.

2.2 Related Work

Here we first compare some representative P2P search techniques. Then, we review some semantic clustering techniques and focus on more details of pSearch.

Comparison of P2P Search Techniques. Table 1 summarizes the search cost and states maintained per peer. In the table, h denotes the average number of nodes in the neighborhood for local indexes; k ($k \leq logN$) denotes the dimensionality for CAN and semantic small world; M and l denote the cluster size and number of long range contacts for semantic small world, respectively. From the comparison, we can see that semantic small world provides flexible tradeoff between states maintained per peer (a constant number of states) and search efficiency (polynomial-logarithmic cost)[6].

Semantic Clustering. The idea of clustering nodes with similar documents together has appeared in [1][4][10][11]. Proposals in [1] and [10] rely on a centralized server or super-peers to cluster documents and nodes. Preliminary work in [4] proposes to cluster nodes with similar interest together, without discussing how to define the interest similarity amongst peers and how to form clusters. [11] relies on periodic message exchanges amongst peers to keep track of other peers with similar documents, which incurs very high message overhead. All these techniques rely on a basic assumption that data objects in a peer are highly homogeneous. On the other hand, while taking advantage of homogeneity in data sets, SSW is suitable for both heterogeneous and homogeneous data sets.

[6] As demonstrated later, the optimal number of long range contacts and size of clusters are small constant numbers.

pSearch. As mentioned earlier, pSearch [15] builds a semantic based information retrieval engine on top of an m-dimensional CAN, where $m = 2.3 \cdot ln(N)$ and N is the total number of nodes selected to support the engine. To address the issues of high dimensionality, pSearch partitions the lower (but more important) dimensions into p groups (where each group consists of m dimensions) and maps the partial semantic space corresponding to each group into the key space of CAN. To process a semantic based similarity search, p separate searches are performed on the CAN key space. The most similar data object(s) in the result of these p searches are returned as the answer. Figure 1 shows an example of pSearch mapping a data object to a 2-dimensional CAN. The example shows that the first six elements of the semantic vector (totally 300 elements) are grouped into three 2-dimensional sub-vectors and mapped to three partial semantic spaces realized in one CAN.

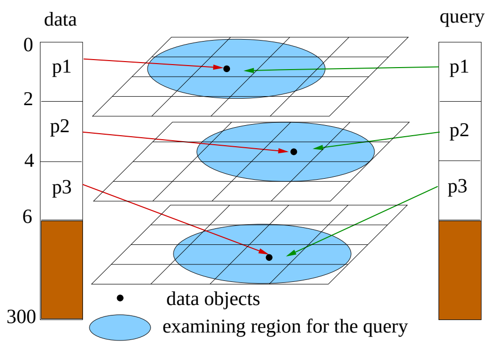

Fig. 1. An illustrative example of pSearch.

In addition to the excessive search and maintenance cost previously mentioned, pSearch has four primary deficiencies. First of all, it lacks the flexibility to accommodate dynamic changes in the system. Since the dimensionality of CAN in pSearch is pre-determined (i.e., m), it is not clear how pSearch can adapt to the changes in network size to achieve the optimal routing hops. Secondly, CAN employs a relatively regular key space partition scheme (i.e., always partition a space evenly) which may result in skewed loads on peer nodes. This deficiency is inherited into pSearch. Thirdly, since only partial semantic information is mapped to CAN, background sampling, which consumes a large amount of bandwidth, especially when peer membership changes frequently, is required to improve the search result. In addition, the effectiveness of this sampling heuristic depends highly on the degree of data object homogeneity stored at a peer. Lastly, the position of a peer in pSearch's semantic spaces does not fully reflect the semantics of its local data objects.

In contrast, our system is much more flexible to the dynamic membership changes. No prior knowledge of the network size is required. Our semantic space partition scheme naturally follows data density, resulting in more fairly distributed load among peers even in skewed data distributions. Since all semantic information, instead of only m dimensions as in pSearch, are used to construct the index/overlay, no heuristics are required to direct the search. Lastly, the placement of peers in the semantic space factors in the semantics of their local data objects, making SSW more adaptive to data locality.

3 Semantic Small World

In this section, we describe the concept of semantic small world (SSW), provide technical details on construction of SSW, and propose a solution to linearize the semantic clusters in high dimensional space into a one-dimensional SSW.

3.1 Overview

SSW serves as both an overlay network and a distributed index structure. Thus, a peer in SSW is not only a storage server of its local data objects but also an index server which provides location information of certain data objects physically stored in other peer nodes. Such location information, similar to the leaf node pointers of typical tree-based index structures, is referred to as *foreign indexes*. Since the data objects are seen as points in a k-dimensional semantic space, peer nodes in SSW are placed in this semantic space using the semantics of their locally stored data objects and are responsible for maintaining foreign indexes for data objects mapped to their semantic subspaces. As a result, peers in SSW are self-organized into *semantic clusters* (with a pre-determined maximum size M) which correspond to semantic subspaces taken charge of by peers within the clusters. The peer nodes within a cluster know each other (directly or indirectly) by keeping track of a pre-determined number (called *out-degree*, where out-degree $\leq M$) of peers within the cluster in a NeighborList. Furthermore, those semantic clusters are self-organized into a small world network for its search efficiency and adaptiveness to dynamics.

Figure 2 shows an example of SSW (with $M = 4, k = 2$). As shown in the figure, the search space is partitioned into 11 clusters after a series of peer joins and leaves. Peer 1 in cluster E maintains short range contacts to neighboring peer clusters A, B and G and a long range contact to a distant peer cluster C. The contacts of other peers are not shown here for clarity of presentation.

3.2 Construction of Semantic Small World

We now discuss how to construct a small world network depicted above. This involves three major tasks: 1) obtaining a *semantic label* that positions a peer node in the semantic space; 2) forming peer clusters in the semantic space; 3) constructing an overlay network across the logical peer clusters to form a semantic small world network.

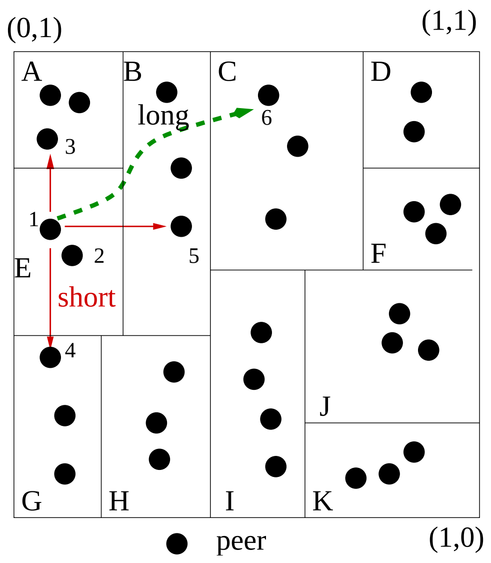

Fig. 2. An illustrative example for SSW overlay structure.

- **Semantic labelling.** This task is executed before or when a peer node joins the network. We assume that each node obtains the SVs for its local data objects by local computation. A peer clusters its local data objects into *data clusters* consisting of data objects with similar semantics [17]. A peer chooses the centroid of its largest data cluster as its semantic label (also called a *join point*) to decide which semantic subspace (and which cluster) the peer is to be placed in. While we assume a single join point here, multiple join points can be used as well. Using centroid of the largest data cluster in a peer node to decide the peer's position in the semantic space has several positive effects. For example, if a node has relatively homogeneous data set (which is likely to be the case in real life), the semantic subspace where a peer resides in is also where most of its data objects fall into, thereby reducing the cost to publish foreign indexes. Moreover, the queries issued by the peers in the nearby subspace usually exhibit similar locality., i.e., a peer is likely to query for data objects with similar semantic meaning as its own data objects. Our construction of SSW exploits these characteristics naturally.
- **Cluster formation.** The SVs of all data objects existing in the system form a virtual search space. Instead of assigning each node to an individual subspace (as CHORD does), SSW is made up of peer clusters (with a preset maximum size M) each capturing a portion of the semantic space. A new peer node joins a cluster which accommodates its semantic label (i.e., join point). If the cluster size exceeds M, the cluster will be split into two based on heuristics (which will be examined later) adopted to maintain good clustering effect. The benefit of this approach is that the size of semantic subspaces

adapts to the density of peers and data objects in the semantic space. In addition, as demonstrated later, clustering makes SSW more stable, and more adaptive to membership changes than other techniques.

– **Overlay network/index construction.** To construct the overlay, each peer node maintains a set of short range contacts pointing to a peer in the neighboring peer clusters and a certain number of long range contacts. The long range contacts are obtained by randomly choosing a point in the semantic search space based on a distribution, $\frac{C}{d^k}$ where k is the dimensionality of the semantic space, d is the semantic distance, and C is a normalization constant that brings the total probability to 1. These extra long range contacts reduce the network diameter and transform the network into a small world with polylogarithmic search cost.

3.3 Dimension Reduction

An intuitive way to construct SSW is to simply assign short range contacts in all dimensions of the semantic space. However, for a semantic space with high dimensionality (e.g., the dimensionality of semantic vector for text documents such as LSI [2] normally is 50-300), this intuitive scheme makes maintenance issues complicated due to the decentralized and highly dynamic nature of P2P systems.

A typical way to address the challenges of high dimensionality is to reduce the dimensionality. However, well-known space filling curves, including the Hilbert curve, Z-curve, Gray-coded curve and Peano curve, cannot be employed naturally for SSW due to the dynamic cluster formation in our proposal that adapts to data density in the semantic space. In this paper, we develop a simple yet effective method, called *adaptive space linearization*, for linearizing the clusters in high dimensional space into a one-dimensional SSW (termed as SSW-1D) through the cluster split process.

When a peer joins the network, if its cluster exceeds the pre-defined maximum cluster size, M, the cluster is partitioned. Our partition strategy is to choose the two peers in the cluster that are semantically farthermost from each other as the seeds for splitting. Then, alternatively assign peers to the two sub-clusters based on the shortest distance to the seeds. Finally, the cluster space is partitioned at the middle point of the dimension that has the largest span between the centroids of the semantic labels of the two sub-clusters (low order dimensions are used to break ties). This is similar to how R-tree nodes are split [3]. Based on this strategy, we obtain two subspaces that have relatively equal number of foreign index entries even though the physical size of the two subspaces may not be equal[7].

Next, we describe a naming scheme which preserves the semantic proximity among clusters as much as possible. To maintain the 1-1 mapping between the naming of clusters in SSW-1D and their semantic subspaces, we use a 128-bit

[7] Existing overlay networks, such as CAN and CHORD, simply partition a space into two equal sized subspaces.

binary number (called *cluster ID*) to name the cluster[8]. Each peer maintains a variable, *Par_Bit*, which initially points to the most significant bit of the cluster ID. Par_Bit indicates the bit to be set (to 0/1) in the next cluster split. After each split, the two sub-cluster decrease their Par_Bit by one (reset Par_Bit to the next less significant bit). Algorithm 1 illustrates the cluster ID encoding process. The first peer cluster in the network sets all bits of its cluster ID to 0. When peers continue to join the network and eventually trigger a cluster splitting, the two sub-clusters obtain IDs by resetting the bit pointed by Par_Bit separately and retaining all other bits the same as the ID of the original cluster. The sub-cluster that has smaller centroid along the partition dimension obtains an ID with the bit pointed by Par_Bit set to 0 and the other one obtains an ID with the bit pointed by Par_Bit set to 1. The same process is employed as more peers join the system to invoke more splits.

Algorithm 1 Algorithm for cluster ID encoding.

Cluster ID encoding when Cluster i with cluster ID $Ci_{128}Ci_{127}...Ci_{1}$ is partitioned into two sub-clusters j and k while j has smaller centroid along the partition dimension. Cj and Ck represent the cluster IDs for Cluster j and k, respectively.

```
1: for $x = 128$ to 1 do
2:   if x = Par_Bit then
3:     $Cj_x = 0$.
4:     $Ck_x = 1$.
5:   else
6:     $Cj_x = Ci_x$.
7:     $Ck_x = Ci_x$.
8:   end if
9: end for
10: $Par\_Bit = Par\_Bit - 1$.
```

Figure 3 shows the process of adaptive space linearization where the whole semantic space is partitioned into 11 clusters with the cluster IDs indicated in the figure. We illustrate the process in a 2-dimensional space where the vertical lines represent the first dimension and the horizontal lines represent the second dimension. We assume that the name space is 4-bit long. In this example, the semantic space is first partitioned along the vertical line as indicated by "p = 1" in the figure. At this point, peers at the left side and right side of this line obtain ID "0000" and "1000", respectively. Then the left side is partitioned along the horizontal line as indicated by "p = 2". At this point, peers at the lower left side and top left side obtain ID "0000" and "0100", respectively. The solid line shows the order of the assigned cluster IDs, while the dashed line indicates that a search can be performed bi-directionally.

[8] We assume the name space is representable by 128-bit IDs.

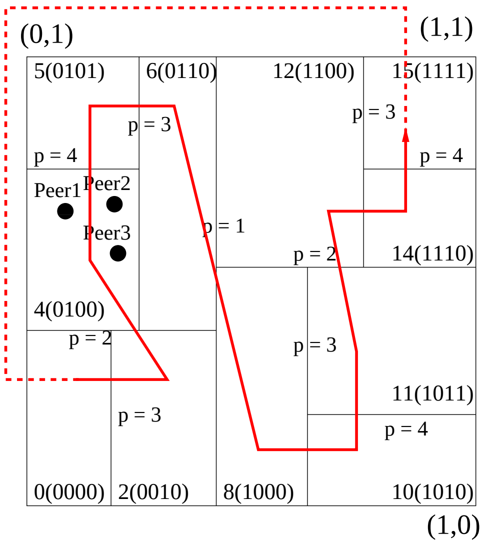

Fig. 3. An illustrative example for adaptive space linearization.

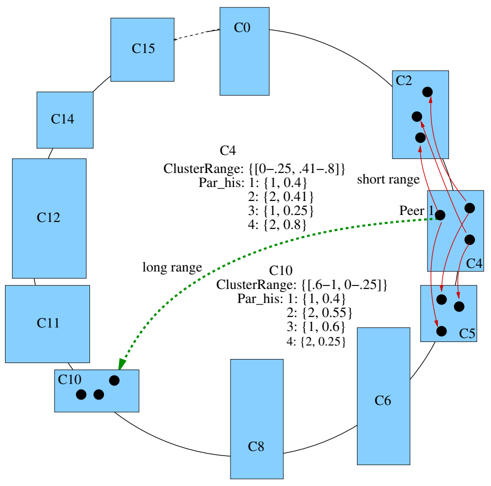

Fig. 4. An illustrative example for SSW-1D.

Figure 4 illustrates SSW-1D built upon the naming scheme described above. A peer in cluster 4 maintains short range contacts to neighboring peer clusters 2 and 5. It also maintains a long range contact to a distant peer cluster 10.

Table 2. Data structure at each peer of SSW

ClusterState: {ClusterRange, ClusterSize, Par_His, Par_Bit}
NeighborList: {NodeId}
ShortContact: {NodeId, ClusterRange}
LongContact: {NodeId, ClusterRange}
ForeignIndex: {Semantic Vector, NodeId}

4 Search and Maintenance

Before we present the search and maintenance operations in SSW-1D, we summarize the information maintained in each peer node (in Table 2). ClusterState consists of ClusterRange, specifying the semantic subspace covered by the cluster this peer node resides in, ClusterSize, indicating the current size of its cluster, Par_His, recording the previous partitions this peer has been involved in, and Par_Bit, indicating the position of next bit to be set for future sub-clusters during next partition. Par_His consists of tuples of ⟨Dimension, Par_Pt⟩ which indicates the partition point along the specified dimension. NeighborList, for intra-cluster search, stores *out-degree* NodeIds of peer nodes within the same cluster. ShortContact and LongContact are self-explanatory. Each contact consists of a NodeId and the ClusterRange of the subspace where the pointed node resides in. ForeignIndex, for the location information of data objects stored at other nodes, consists of a set of semantic vectors of data objects as well as the NodeIDs of their source nodes.

4.1 Search

To initiate a content-based search, a requester first generates a *search semantic vector* (denoted as SV_s) based on the query terms. Here we focus on the search process, which consists of the *flooding search* and *navigation* stages, to reach the node holding foreign indexes (or the data itself) for data objects satisfying the query. Correspondingly, the search operation at a peer node has two modes: *search-within-cluster* and *search-across-cluster*. Algorithm 2 illustrates the search process. When a message is received, a peer node will first check whether SV_s falls within the range of its cluster. If that is the case, it starts search-within-cluster mode by flooding the message to peers in its NeighborList (except for the one from whom the message was received[9]). Then the data object with highest similarity to the query is returned as the result. Otherwise, the search-across-cluster mode is invoked. A pseudo-cluster-name (PCN), the estimated cluster name for the semantic subspace where the query resides in, is calculated for SV_s based on the partition history (Par_His) stored at this peer. First, we set all the bits of PCN to be 0. Iterating through the Par_His of the peer, the bits

[9] A sequence number is attached to each search message so that a node can recognize and drop a search message that appeared before.

of PCN are set as the same value of corresponding bits of this peer's Cluster ID as long as the SV_s confirms to the same Par_His entry. Otherwise, the PCN resolving process at this node stops and the corresponding bit is set to a different value since this peer does not have further details about the PCN. The search-across-cluster mode is initiated/continued by forwarding the search message to the contact with the shortest naming distance to the PCN. The above process is repeated until the cluster whose semantic subspace covering SV_s is reached.

Algorithm 2 Algorithm for Search.

```
Search at Peer i: $i.search(SV_s)$
 1: if $SV_s \in i.ClusterRange$ then
 2:    if receive $search(SV_s)$ before then
 3:       drop $search(SV_s)$.
 4:    else
 5:       forward $search(SV_s)$ to all members in its NeighborList excluding the imme-
          diate sender.
 6:    end if
 7: else
 8:    Calculate PCN for SV$_s$.
 9:    $m$ = closest contact to PCN of Peer $i$.
10:    forward $search(SV_s)$ to $m$.
11: end if
```

Here, we show an example to illustrate the search process in SSW-1D. Let's go back to Figure 4. Assuming that Peer 1 in Cluster 4 wants to search for data objects based on SV_s [0.9,0.3], it first checks its own cluster range. Since [0.9,0.3] is not within the subspace of the cluster, Peer 1 then calculates the PCN for the query. It starts from the first entry in its Par_His. The first partition is at coordinate 0.4 along the first dimension while Cluster 4 is at the left half ([0-0.4]). Therefore, the first bit of the query's PCN is set to 1 and the PCN resolving process stops with a PCN=8 for the query. Then Peer 1 checks its contacts and forwards the search to the closest peer node (in the naming space), which is a peer in Cluster 10 in this example. When the query reaches a peer in Cluster 10, this peer re-calculates the PCN for the query since the query is still not in its cluster range. This time, the query PCN is resolved as "1011" (i.e., Cluster 11). The search is finally forwarded to a peer in Cluster 11, which finds SV_s within its own cluster range, so it floods the cluster for results.

4.2 Peer Join

Peer join needs to be handled properly in order to keep maintenance costs low while adhering to the effectiveness and efficiency of the SSW. In addition to the obvious task of locating and joining a cluster, there are two other crucial tasks: 1) *cluster splitting*, and 2) *foreign index publishing*.

- **General Process.** Let *x.label* denote the semantic label of the join point chosen by Peer x. Peer x joins the network by sending a join message with this join point to an existing Peer i in the network (a list of existing peers are known to new peers). Peer i first performs a search and directs the join message to one of the peer nodes (e.g., Peer j) in the cluster that covers the SV of the join point. This peer node is called a *contact peer*. If the cluster size is below the maximum cluster size M, Peer x simply joins the cluster. The membership changes are disseminated to other members during routine message exchanges, such as search messages, without additional communication cost.

- **Cluster Splitting.** If the cluster size exceeds M, cluster splitting is invoked by the contact peer. The contact peer first obtains a complete list of the semantic labels of all peers in the cluster by polling the peers in the cluster through flooding. Then it splits the cluster into two according to the cluster splitting strategy described in Section 3.3. The contact peer finishes the cluster splitting by informing all peers in its cluster to update their ClusterState. In addition, peers transfer the foreign indexes no longer belonging to their new sub-clusters to the other sub-clusters. A peer also updates its inter-cluster state (e.g., replacing the ShortContact) and informs the neighboring peer clusters. The cluster splitting operation is invoked infrequently, i.e., only when the number of active nodes in a cluster exceeds M. Large M should bring down the maintenance overheads since less number of splits are conducted. However, the search cost will increase due to flooding within a cluster. The effect of M will be evaluated in the simulation later.

- **Foreign Index Publishing.** After joining a cluster, a peer may find that some of its local data objects do not belong to this cluster. As a result, the newly joined peer publishes the locations of these data objects to their corresponding peer clusters (as foreign indexes). The first node (in a corresponding peer cluster) reached during the publishing process adds a tuple consisting of SV and the NodeID of source node for a data object into its foreign index store.

4.3 Peer Leave

When a peer leaves the network, it checks whether it is the last peer in the cluster. If there are other peers in the cluster, this peer simply informs its leaving by transferring its foreign index to a randomly selected peer in its cluster. Otherwise, the semantic subspace of this cluster needs to be merged with one of its neighboring clusters. To perform a merge, a leaving peer first transfers the foreign index to a selected neighboring cluster. The peer receiving the transferred index updates its cluster range as well as the affected short range contacts. Similarly, this change is attached with other routine messages so that other peers in this newly merged peer cluster update their cluster range as well as the affected short range contacts.

4.4 Peer Failure

A failed peer is detected during routine operations such as search. If a peer detects a failure in one of its long range contacts, it re-establishes this contact simply as explained in Section 3.2. On the other hand, if the peer detecting a failure is located in one of the neighboring clusters E of the failed peer, originally located in cluster F, it is likely that other peers in cluster E maintain short range contacts to other live peers in cluster F. Therefore, at the expense of two messages, the short range contact of the detecting peer can be recovered. If a short range contact of a peer in cluster E cannot be recovered by contacting other peers in its cluster, it implies that no live peer exists in cluster F. At this point, cluster merging as described for peer leave could be initiated at cluster E.

5 Performance Evaluation

We move on to evaluate SSW's benefits using simulations. We compare SSW-1D (we refer it as SSW in this section for simplicity) with pSearch, the state-of-the-art in semantic-based P2P search. The goal of a search is to find a data object semantically similar to the query specified. Based on [15], pSearch takes 4 groups of the most important dimensions, each with m dimensions (i.e., $p = 4$, $m = 2.3 lnN$). The simulation setup, parameters and performance metrics are explained below.

5.1 Simulation setup

The simulation is initialized by having one node pre-exist in the network and then injecting node join operations into the network till the network reaches a certain size (N). After this point, a mixture of operations including peer join, peer leave and search are randomly (based on certain ratios) injected into the network. This is also when statistics collection begins. On the average, each peer issues 100 searches during each run of the simulation. The proportion of join to leave operations is kept the same to maintain the network at approximately the same size. The simulation parameters, their values and the defaults (unless otherwise stated) are given in Table 3. Most of these parameters are self-explanatory. More details for some of the parameters are given below.

Increasing the number of long range contacts should bring down the search path length, but the maintenance cost will increase. Increasing cluster size should decrease the maintenance cost at the expense of high search cost within a cluster due to flooding. These tradeoff are evaluated in detail. Without loss of generality, the dimensionality of SV (and the semantic space) is set to 100 and the dataset used in the simulation follows random distribution.

5.2 Metrics

We use the following metrics for our evaluations:

Table 3. Parameters used in the simulations

	Descriptions	**Values, default**
N	Number of nodes in the network	256 - 16K, 1K
l	Number of long range contacts	1 - 6, 4
M	Size of peer clusters	1 - 1024, 8
x	Out-degree within peer clusters	4
n	Number of data objects per peer	1 - 100, 100
γ	Percentage of join/leave operations	0% - 50%, 20%

- **Search path length** is the average number of logical hops traversed by search messages to the destination.
- **Search cost** is the average number of messages incurred per search. Flooding techniques like Gnutella may have short path length, but their search cost is high.
- **Maintenance cost** is the number of messages incurred per membership change, consisting of **overlay maintenance cost** and **foreign index publishing cost**. In some cases, we also use **cost per operation** to denote the amortized number of messages per operation. Cost per operation is calculated by dividing the total number of messages incurred by all searches and other operations causing membership changes (e.g., peer join, leave) by the total number of operations.

6 Simulation Results

In this section, we first demonstrate the scalability of SSW in terms of the size of the network and the number of data objects in the system. We then examine the effect of cluster sizes on SSW.

6.1 Scalability

In terms of scalability to network size, we vary the number of nodes from 2^8 to 2^{14} to evaluate the search efficiency and maintenance cost of SSW with different numbers of long range contacts (i.e., 1-6). Since pSearch does not use any clustering, we disable the clustering feature of SSW (i.e., cluster size is set to 1) in these experiments. A later experiment will evaluate SSW with various cluster sizes and show that it can perform even better with appropriate cluster sizes.

Figure 5 shows the average path length when we vary the number of long range contacts from 1 to 6. Since the size of peer clusters is set to 1 in this experiment, there is no flooding within a cluster and thus the average search path length for SSW represents the search cost as well. The search path length for SSW increases slowly with the size of network. The slope of pSearch's path length is close to SSW with 4 long range contacts but with a much higher offset.

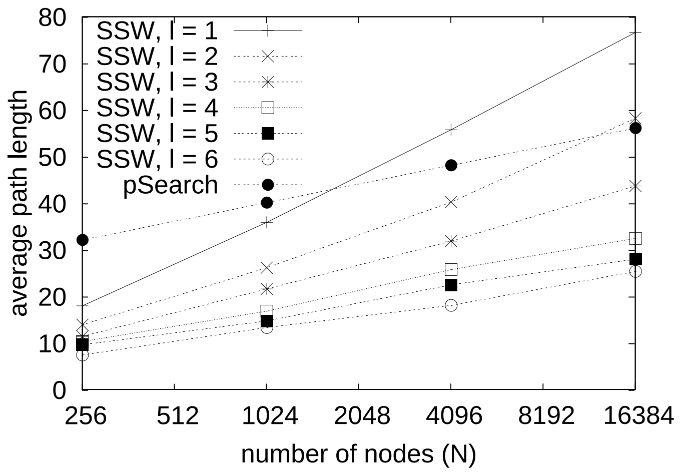

Fig. 5. Comparing the network size scalability of the schemes with respect to search path length. Results for SSW are shown with different number of long range contacts.

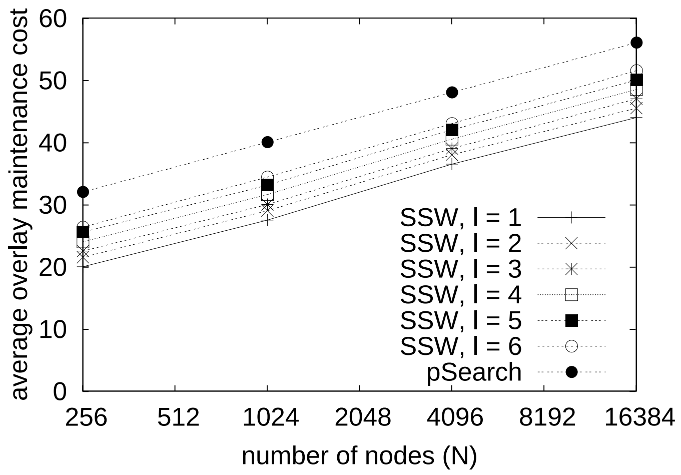

Fig. 6. Comparing the network size scalability of the schemes with respect to overlay maintenance cost.

Overlay maintenance cost is proportional to the number of states maintained at each peer, which are 20 and $2 + l$ (2 short range contacts and l long range contacts) for pSearch and SSW respectively. Figure 6 shows the overlay maintenance cost for the same experiments as Figure 5. These two figures confirm our expectation that compared to pSearch, SSW can achieve better search performance with much smaller number of states maintained per peer.

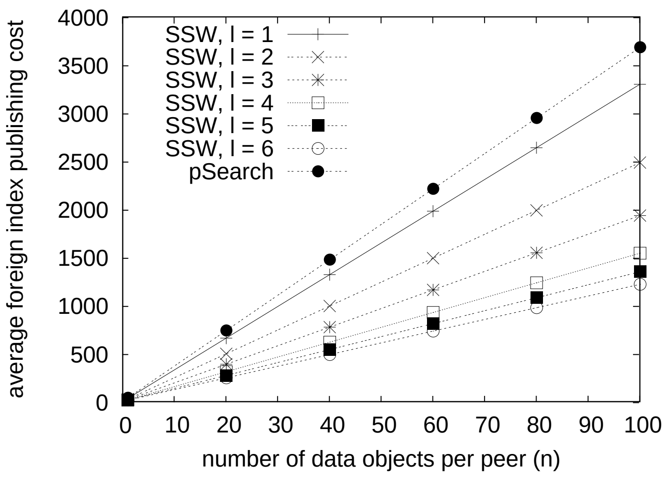

Fig. 7. Comparing the foreign index publishing costs as a function of the number of data objects per peer.

The other maintenance cost to consider is the overhead of publishing foreign index at peer joins (apart from the cost shown in Figure 6). This cost is proportional to the number of data objects that need to be published, and the corresponding relationship is shown in Figure 7 for different long range contacts. Due to the fact that pSearch has to publish a data object multiple times, the index publishing cost for pSearch is much higher than SSW.

Having considered the costs of individual operations, it is important to put these in perspective to understand the trade-offs between search efficiency and the maintenance overheads. This is obviously a function of the proportion (γ) of different operations that are being considered. In Figure 8, we show the average cost per operation for different values of γ on the x-axis. In the graph, the bars for pSearch (the last bar) are given for easier comparison.

The number of long range contacts (l) for SSW can be tuned according to network stability. For instance, if a network is relatively stable with infrequent membership changes, the number of long range contacts can be increased to reduce search path length. In other environments where membership changes frequently, long range contacts can be limited to a small number to keep the maintenance costs low (these are again confirmed by the results in Figure 8). We find that 4 long range contacts is a reasonable trade-off point between search efficiency and maintenance overhead, and we use this value in next experiment.

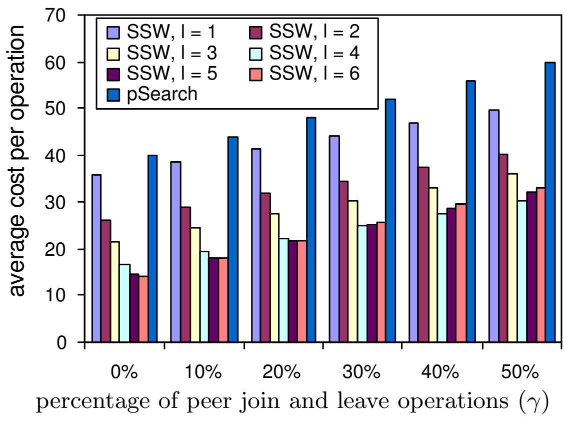

Fig. 8. Effect of varying γ, the percentage of Join/ Leave under different number of long range contacts.

6.2 Clustering Effects

Until now the size of the peer cluster (M) has been set at 1. When M is larger, cluster splits or merges occur less frequently, resulting in lower overlay maintenance costs. Further, the total number of clusters in the system decreases with larger cluster sizes, thereby reducing searches across clusters. The down-side of large sized clusters is the higher search cost within a cluster (due to flooding).

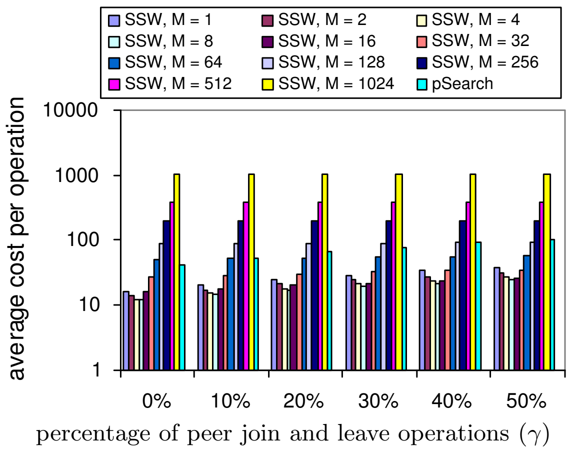

Fig. 9. Effect of Varying γ, the percentage of Join/Leave operations under different cluster size.

The effect of the cluster size on cost per operation are given in Figure 9. The cluster size is varied between 1 (the size used in the simulations until now) and 1024 (the whole network is one big cluster). We also vary the percentage of join/leave operations in this set of experiment. From this figure, we can see that within a spectrum of cluster sizes between 2 and 16, SSW does better than the size of 1 (whose results were presented in the previous section) in terms of cost per operation.

Similarly, the size of peer clusters can be tuned to an optimal point according to application requirements and network stability. The size of peer clusters can be increased to reduce maintenance overhead under unstable environments where peers join or leave the network frequently, and can be decreased to make search more focussed under stable environments.

7 Conclusion

Peer-to-Peer 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 search efficiency and network traffic while being scalable to network size and data volume. In this paper, we propose an overlay network, namely, semantic small world (SSW), to support semantic based search in peer-to-peer systems. Peers in SSW reside in a semantic space in accordance with the semantics of data objects stored locally and form clusters with other peers residing in the same semantic subspace. These peer clusters are self-organized into a small world network which has efficient search performance with low maintenance overhead. To address the issues of high dimensionality, we proposed a dynamic dimension reduction method, called adaptive space linearization, to map a high dimensional semantic space to a one-dimensional SSW. SSW facilitates efficient search without incurring high maintenance overhead. By placing and clustering peers in the semantic space based on the semantics of their data objects, SSW adapts to distribution of data automatically. With these attractive features, we believe that SSW can have a significant impact on the deployment of large scale P2P applications.

We are currently conducting more in-depth simulations and evaluating effects of various factors, such as skewness of data distribution, node failure rate, etc., on the performance of SSW. We are also exploiting strategies for various types of searches, such as K nearest neighbor search and partial lookup. We are exploiting resource heterogeneity amongst peers by dynamically adjusting number of join points, long range contacts and cluster size. We also plan to investigate locality of interest in multiple queries.

References

1. M. Bawa, G. S. Manku, and P. Raghavan. SETS: Search enhanced by topic segmentation. In *Proceedings of ACM SIGIR*, pages 306–313, July 2003.

2. M. W. Berry, Z. Drmac, and E. R. Jessup. Matrices, vector spaces, and information retrieval. *Society for Industrial and Applied Mathematics Review*, 41(2):335–362, 1999.
3. A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of ACM SIGMOD*, pages 47–54, 1984.
4. A. Iamnitchi, M. Ripeanu, and I. T. Foster. Locating data in (small-world?) peer-to-peer scientific collaborations. In *2nd International Workshop on Peer-to-Peer Systems (IPTPS)*, pages 232–241, March 2002.
5. J. Kleinberg. Navigation in a small world. *Nature*, 406(845), August 2000.
6. J. Kleinberg. The small-world phenomenon: an algorithm perspective. In *Proceedings of ACM Symposium on Theory of Computing*, pages 163–170, May 2000.
7. M. Li, W.-C. Lee, and A. Sivasubramaniam. Neighborhood signatures for searching P2P networks. In *Proceedings of International Database Engineering and Application Symposium (IDEAS)*, pages 149–158, July 2003.
8. Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *Proceedings of ACM International Conference on Supercomputing*, pages 84–95, June 2002.
9. G. S. Manku, M. Bawa, and P. Raghavan. Symphony: Distributed hashing in a small world. In *Proceedings of USENIX Symposium on Internet Technologies and Systems*, March 2003.
10. W. Nejdl, M. Wolpers, W. Siberski, C. Schmitz, M. T. Schlosser, I. Brunkhorst, and A. Lser. Super-peer-based routing and clustering strategies for RDF-based peer-to-peer networks. In *Proceedings of International World Wide Web Conference (WWW)*, pages 536–543, May 2003.
11. C. H. Ng, K. C. Sia, and C. H. Chang. Advanced peer clustering and firework query model in the peer-to-peer network. In *Proceedings of International World Wide Web Conference (WWW), Poster*, May 2003.
12. 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.
13. A. I. T. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, November 2001.
14. 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.
15. C. Tang, Z. Xu, and S. Dwarkadas. Peer-to-peer information retrieval using self-organizing semantic overlay networks. In *Proceedings of ACM SIGCOMM*, pages 175–186, August 2003.
16. B. Yang and H. Garcia-Molina. Improving search in peer-to-peer networks. In *Proceedings of International Conference on Distributed Computing Systems (ICDCS)*, pages 5–14, July 2002.
17. T. Zhang, R. Ramakrishnan, and M. Livny. Birch: An efficient data clustering method for very large databases. In *Proceedings of ACM SIGMOD*, pages 103–114, June 1996.
18. 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.