On Constructing Small Worlds in Unstructured Peer-to-Peer Systems* **

Yannis Petrakis and Evaggelia Pitoura

Department of Computer Science, University of Ioannina, Greece.
{pgiannis, pitoura}@cs.uoi.gr

Abstract. Peer-to-peer systems have evolved as a means to share large amounts of data among autonomous nodes. A central issue in this context is locating nodes with data matching a user query. In this paper, we consider building peer-to-peer systems with small-world properties, that is, connecting the nodes to each other so that: (i) the distance between any two nodes is small and (ii) relevant nodes are connected to each other. Relevance between nodes is defined based on the probability that the two nodes match similar queries. We propose decentralized procedures for constructing small-worlds based on routing indexes that describe the content of neighboring nodes. Our experimental results show that small-world peer-to-peer systems built with these procedures increase recall, that is, the percentage of relevant results returned.

1 Introduction

The popularity of file sharing systems such as Napster, Gnutella and Kazaa has spurred much current attention to peer-to-peer (P2P) computing. Peer-to-peer computing refers to a form of distributed computing that involves a large number of autonomous computing nodes (the peers) that cooperate to share resources and services [12]. A central issue in P2P systems is identifying which peers contain data relevant to a user query.

In this paper, we propose building small worlds based on the content of the peers. Small worlds are networks with (i) a small distance among any two nodes (small diameter) and (ii) a large number of connections among relevant nodes (large clustering coefficient) [19]. We define the relevance of two nodes (peers) based on the probability of them matching the same set of queries. Intuitively, the topology of a small world network represents a number of smaller networks (groups) that are rich in links between their peers (short-range connections), while they are linked to each other with a few random connections (long-range connections). The motivation for such small-world P2P networks is that once in the appropriate group, all relevant to a query peers are a few links apart. Long-range links are used for routing among groups.

* In the EDBT International Workshop on Peer-to-Peer Computing and Databases, Heraklion, Crete, Greece, March 14, 2004

** Work supported in part by the IST programme of the European Commission FET under the IST-2001-32645 DBGlobe project

We present an approach for building small worlds based on a fully decentralized procedure. Our construction is based on using local indexes. A local index is a characterization of the content of a peer. By aggregating local indexes of neighboring nodes, we create small worlds in a fully distributed manner.

To demonstrate our approach, we implemented routing indexes using Bloom filters [3]. Bloom filters are bit vectors used for probabilistic representation of a set to support membership queries. Our performance results show that networks constructed by our procedures have the small world properties. Moreover, they maximize recall and precision, that is, for a given query they increase the number of matching data returned while maintaining the number of peers visited small.

The remainder of this paper is structured as follows. In Section 2, we put our work in context with related research. In Section 3, we present the model of our system, while in Section 4, we describe how small worlds are built. In Section 5, we present experimental results. Section 6 concludes the paper.

2 Related Work

There are two basic types of P2P systems: structured and unstructured ones. In *structured P2P* systems, documents (or indexes of documents) are placed at specific nodes (peers) usually based on distributed hashing (DHTs) such as in CAN [13] and Chord [6]. With distributed hashing, each document is associated with a key and each peer is assigned a range of keys and thus documents. Peers are interconnected via a regular topology where peers that are close in the identifier space are highly interconnected. Very recently, researchers have proposed extending DHTs (e.g., Chord) with long range links towards creating small worlds [11]. In addition, recent extensions propose instead of associating keys to documents based on just the identifier of the document, to associate with each document (or peer) a vector describing its content extracted using IR algorithms and then use this vector as input to the hashing functions [16, 15]. However, this creates a dimensionality reduction problem, since the dimension of the vectors should match the dimension of the DHT.

These proposals can collectively be seen as an approach of building content-based small worlds in DHT-based P2P systems. In this case, the usual problems with structured P2P systems arise, since although DHTs provide very efficient searching, they compromise peer autonomy. The DHT topology is regulated since all peers have the same number of neighboring peers and the selection of peers is strictly determined by the DHTs semantics. Furthermore, sophisticated load balancing procedures are required.

We propose building small-worlds in unstructured (non DHT-based) P2P systems. Unstructured P2P systems can be further distinguished between systems that use indexes and those that are based on flooding and its variations. With flooding (such as in Gnutella), a peer searching for a document contacts its neighbor peers which in turn contact their own neighbors until a matching peer is reached. Flooding incurs large network overheads. In the case of indexes, these can be either centralized (as in Napster), or distributed among the peers (as in

routing indexes [4]) providing for each peer a partial view of the system. We show how by using such indexes, we can organize the peers in small-worlds in a fully decentralized manner. Small-worlds in non DHT P2P systems are also discussed in [1] in the context of searchable querical data networks; however, this work does not include a concrete decentralized small-world construction procedure.

Finally, many recent research efforts are focusing on organizing peers in clusters which in a sense are similar to groups in small-worlds. In most cases, the number or the description of the clusters is fixed and global knowledge of this information is required. In [2], peers are partitioned into topic segments based on their documents. A fixed set of C clusters is assumed, each one corresponding to a topic segment. Knowledge of the C centroids is global. Clusters of peers are formed in [18] based on the semantic categories of their documents; the semantic categories are predefined. Similarly, [5] assumes predefined classification hierarchies based on which queries and documents are categorized. The clustering of peers in [10] is based on the schemes of the peers and on predefined policies provided by human experts. Besides clustering of peers based on content, clustering on other common features is possible such as the formation of communities of peers based on their interests [8].

3 Content-Based Small Worlds in P2P Systems

We assume a P2P system with a set N of peers n_i. Each peer stores a set of data items (such as documents or relations). Each peer is connected to a small number of other peers called its *neighbors*. A query q may be posed at any of the peers. We denote by $match(n_i, q) = true$ the fact that peer n_i has data items satisfying a query q; otherwise $match(n_i, q) = false$. Peers with data satisfying the query are called *matching* peers. In the following, we use the notation $|S|$ to denote the number of elements of a set S.

3.1 Small Worlds in P2P Systems

The *distance* between two peers n_i and n_j, $dist(n_i, n_j)$ is the length of the shortest path from n_i to n_j. The *diameter* of the network is the maximum distance between any two peers in the network. The *clustering coefficient* of a network captures the probability that two neighbors of a peer are also neighbors themselves; it is the average fraction of pairs of neighbors of a peer that are neighbors of each other. Small world networks are characterized by: (i) a small diameter and (ii) a large clustering coefficient [19].

Intuitively, the topology of a small world network represents a number of smaller networks that are rich in links between their peers (short-range connections) and these smaller networks are linked between them with a few connections (long-range connections). The small world phenomenon finds many applications in real life [19, 7]. Friendship networks are a good example of this. Consider the friendship graph, where each peer corresponds to a person and two people are connected with an edge if they know each other. Such a graph consists of smaller

sub-graphs (which are rich in short-range connections) each one representing a community and there are a few long-range links between peers of different communities (if A knows B and B knows C, then A is more likely to know C than some other random person).

Our goal is to organize peers into groups based on some characteristics, where there are many structured inter-group (short-range) links and a few random intra-group (long-range) links. The short range links are created based on a common characteristic of the peers that are connected, while the long range links are constructed randomly between peers of different groups. These links serve to reduce the average path length between any two peers in the network.

More specifically, we want to group together the peers that match the same set of queries. The motivation for building such small worlds in P2P systems is to increase the number of matching peers that are returned. This is because, if peers that match similar queries are linked together, once we find one matching peer, all others are nearby. In particular, let $Visited$ be the set of peers visited while processing a query and $Matching$ be the set of all peers in N that match the query. Then, $Recall = (|Matching \cap Visited|)/|Matching|$ and $Precision = (|Matching \cap Visited|)/|Visited|$. Ideally query propagation should visit only peers in the $Matching$ set.

In particular, we want the probability that n_i and n_j are neighbors to depend on their similarity. We define the similarity among two peers based on the probability that they match the same set of queries:

Definition 1. (similarity among peers) *For two peers n_i and n_j, we define their similarity as follows: $psimilar(n_i, n_j) = min(a_1, a_2)$ over all queries q, where $a_1 = Probability\{match(n_i, q) \Rightarrow match(n_j, q)\}$ and $a_2 = Probability\{match(n_j, q) \Rightarrow match(n_i, q)\}$.*

Figure 1 shows a ramdom and a small-world P2P network. In a small-world network, peers that match a query are nearby.

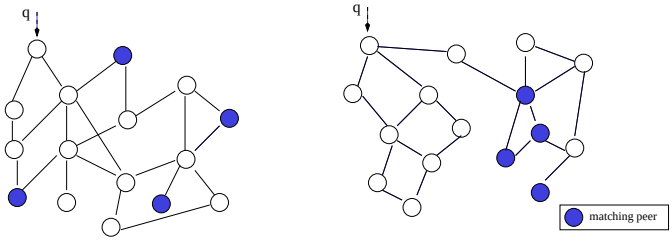

Fig. 1. (left) Random and (right) small world p2p network

3.2 Content Summaries

Our approach is based on using local indexes to describe the content of each peer. We take the following generic approach: we assume that there is an "index" or

summary called a local index $LI(n)$ that describes the content of each peer n. For instance, the index may be an inverted list of keywords, a B-tree, or a Bloom filter (as in our case study). A property of the "index" is that we can determine, with high probability, whether the peer satisfies the query based on the index of the peer, that is, without looking at the actual content of the peer. In particular, each index supports a predicate *imatch* such that: $\forall$ query q and peer $n \in N$, if $match(n, q) = true$ then $imatch(LI(n), q) = true$. The inverse also holds with a high probability. The case in which the inverse does not hold is called a *false positive*.

Besides its local index, each peer n maintains one routing index $RI(n, e)$ per link e, that describes the content of all peers that are reachable from n using link e at a distance at most R.

Definition 2. (horizon and radius) *A peer n_i has a horizon of radius R, if for each edge e, it maintains a routing index $RI(n_i, e)$ such that: $\forall$ query q and $n_j \in N$, such that if using link e, $dist(n_i, n_j) \leq R$ and $match(n_j, q) = true$ then $imatch(RI(n_i, e), q) = true$.*

The type of index depends on the type of data items of each peer that we want to share (e.g., simple files, relational databases, XML files) as well as on the type of queries (e.g., keyword-based, SQL-like or path queries). The index is representative of the content. In particular, we define a similarity metric among indexes, denoted as *similarity*, such that the more similar the peers, the more similar their indexes.

4 Building and Querying Small Worlds

Our goal is to create groups of similar peers. We distinguish between two types of links: *short-range* or short links that connect similar peers and *long-range* or long links that connect non-similar peers. Two peers belong to the same *group* if and only if there is a path consisting only of short links between them.

4.1 Query Routing

Our goal is to locate all matching peers. Assume that a query q is posed at peer n. First, n checks whether its own local index matches the query (whether $imatch(LI(\text{n}), q) = true$). Then, it propagates the query only through one or more of those links e whose routing indexes match the query ($imatch(RI(n, e), q) = true$). Analogously, each peer that receives the query first checks its own local index and propagates the query only through those links whose routing indexes match the query. This procedure ends when a maximum number of peers ($MaxVisited$) is visited or when the desired number of matching data items (results) is attained.

Note that following this procedure, it is possible to reach a situation in which no matching peers are found. This can happen for example if the peer n that poses a query has no matching outgoing links ($imatch(RI(n, e), q) = false$ $\forall$

link e of n), which means that the matching peers (if any) are outside the radius R of n. To handle this case, we use a variation of the above procedure until we find the first matching link. Specifically, if no matching link has been found yet during the routing of the query, and the current peer has no matching outgoing links, then the long link of this peer is followed (even if it does not match the query). The idea is that we want to move to another group of the network, since the current group (bounded by the horizon) has no matching peers. In the case that the peer has no long link or we have already followed all long links, the query is propagated through a short link to a neighbor peer and so on until a long link is found.

4.2 Small World Construction

Each new peer that enters the system tries to find a relevant group of peers and links with SL of them through short links. Also, it may link with a peer that does not belong to this group through a long link. Short links are inserted so that the peers with relevant information are located nearby and a large clustering coefficient is attained. Long links are used for keeping the diameter small. The idea is that we want to be easy to find both all the relevant results once in the right group, and the relevant group once in another group, thus reducing the number of hops that are needed to answer a query.

Specifically, when a new peer n joins the system, a join message that contains its local index $LI(n)$ is posed as a query to a well known peer in the system. This message maintains two lists (initially empty): (i) a list with peers of the same group with peer n, denoted *clist*, and (ii) a list with peers of different groups, denoted *dlist*. Whenever the message reaches a peer p, the similarity between the local indexes $LI(n)$ and $LI(p)$ is calculated. If the similarity is below a threshold t ($similarity(LI(n), LI(p)) < t$), peer p is added to the list *dlist* which means that p and n should belong to different groups. Otherwise, p is added to *clist*.

The join message is propagated using a query routing procedure that exploits the routing indexes as follows. The similarity between the local index $LI(n)$ of the new peer n and the routing indexes $RI(p, e)$ that correspond to each of the outgoing links e of peer p is calculated. The message is propagated through the most similar link e (for which $similarity(RI(p, e), LI(n))$ is the maximum) because there is higher probability to find the relevant group through this link. This routing procedure is followed until either SL peers n_i with $similarity(LI(n_i), LI(n)) \geq t$ are found or a predefined number of peers (denoted $JMaxVisited$) is visited. Then a peer u is selected from the list *dlist*. The new peer connects to the peers of *clist* through short links and to peer u through a long link. If *dlist* is empty, which means that the message has not reached any peer from another group, the join message is also routed through the link for which the probability to reach a peer with similarity below the threshold t is the highest. That is, the message is propagated through the link e for which the routing index is less similar with the local index of the new peer ($similarity(RI(p, e), LI(n))$ is the minimum). The procedure stops when such a peer is found.

One disadvantage of the above algorithm is that it is difficult to define an appropriate value for the threshold. Furthermore, this value may need to change as the content of the peers changes. To address these issues, we also consider the following procedure that does not use threshold: the routing of the join message continues until the maximum number of peers visited $JMaxVisited$ is reached. The message is always propagated through the link that is most similar with the local index of the new peer. The message maintains a list $MaxMinList$ of all the visited peers and the corresponding similarities between their local indexes and the local index of the new peer. When the routing stops, the new peer selects to be linked to the SL most similar peers in $MaxMinList$ through short links, and randomly to one of the other peers in the list through a long link. We call the algorithm that uses a threshold, *threshold algorithm*, and the algorithm that does not use threshold *MaxMin algorithm*.

5 Experimental Evaluation

5.1 Bloom Filters as Routing Indexes

Bloom filters are compact data structures for probabilistic representation of a set that support membership queries ("Is element a in set A?"). Consider a set $A = \{a_1, a_2, ..., a_n\}$ of n elements. The idea is to allocate a vector v of m bits, initially all set to 0, and then choose k independent hash functions, h_1, h_2, ..., h_k, each with range 1 to m. For each element $a \in A$, the bits at positions $h_1(a)$, $h_2(a)$, ..., $h_k(a)$ in v are set to 1 (Fig. 2). A particular bit may be set to 1 many times. Given a query for b, the bits at positions $h_1(b)$, $h_2(b)$, ..., $h_k(b)$ are checked. If any of them is 0, then certainly $b \notin A$. Otherwise, we conjecture that b is in the set although there is a certain probability that we are wrong. This is a false positive. It has been shown [3] that the probability of a false positive is equal to $(1 - e^{-kn/m})^k$.

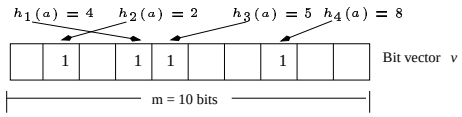

Fig. 2. A Bloom filter with $k = 4$ hash functions

Bloom filters have been used as routing indexes in P2P systems [14, 9]. The routing index $RI(n, e)$ of radius R of the outgoing link e of peer n is equal to the bitwise OR of the local indexes of all peers p within distance R of n reachable through link e. Let B be a Bloom filter of size m. We shall use the notation $B[i]$, $1 \leq i \leq m$ to denote the ith bit of the filter. Let two Bloom filters B and C of size m, their Manhattan (or Hamming) distance, $d(B, C)$ is defined as $d(B, C)$ = |B[1] - C[1]| + |B[2] - C[2]| + ... + |B[m] - C[m]|, that is the number of bits that they differ. The smaller the distance, the more similar the Bloom filters.

5.2 Simulation

We associate a document with each node. Each document has a number of elements (keywords). Every 10% of the documents are 50% similar to each other in terms of elements in common. For the hash functions, we use MD5 [17] that is a cryptographic message digest algorithm that hashes arbitrarily length strings to 128 bits. The k hash functions are built by first calculating the MD5 signature of the input string, which yields 128 bits, and then taking k groups of 128/k bits from it. The filter size is set to 6000 bits with 4 hash functions. The size of the network is set to 500 peers and the horizon radius varies from 1 to 3. Each new peer creates one short link ($SL = 1$) and one long link with probability $P_l = 0.4$. The routing of the join message stops when a maximum number ($JMaxVisited$) of peers has been visited, varying from 5% to 20% of the existing peers. Analogously, the routing of the query stops when a maximum number ($MaxVisited$) of peers has been visited. This number remains constant to 10% of the network peers. In the first set of experiments, we show that our network satisfies the small world properties. In the second set of experiments, we examine the performance of such a network, considering as performance metrics: (i) recall (i.e., the percentage of results found) and (ii) the number of hops until the first result is found. Table 1 summarizes our input parameters

Table 1. Input parameters

Parameter	Default Value	Range
Number of peers	500	
Radius of the horizon	3	1-3
Number of short links (SL)	1	
Probability of long link (P_l)	0.4	
Percentage of peers visited during join ($JMaxVisited$)	10	5-20
Percentage of peers visited during query ($MaxVisited$)	10	
Number of hits per peer	1	
Filter size	6000bits	
Number of hash functions	4	
Number of elements per document	80	
Percentage of hits (matching peers per query)	7%	

In the following, we examine the influence of the *join radius* (i.e., the radius used during the creation of the network). Note that we vary the radius used by a peer when joining the network, while we keep the radius used in query processing constant and equal to 3. The reason is to show that the radius used during the construction of the network affects the quality of the constructed small-world network. We also vary the percentage of peers visited during the join procedure ($JMaxVisited$). We evaluate both the $MaxMin$ and the $threshold$ algorithms and compare the constructed small-world networks with a random one.

5.3 Network Properties

We study the properties of the networks created using the two algorithms, in particular, their diameter and degree of clustering.

Diameter We calculate the diameter of the network constructed following our procedures for creating a small world (Fig. 3). Our goal is to show that the network we build satisfies the small world property of a small diameter O(log$N$) where N is the number of peers. We calculate the diameter for different values of the join radius. The value of the diameter is about 10 which means that it satisfies the property. Also, the diameter of the networks created using the *threshold* and $MaxMin$ algorithms is a little smaller than the diameter of the random network.

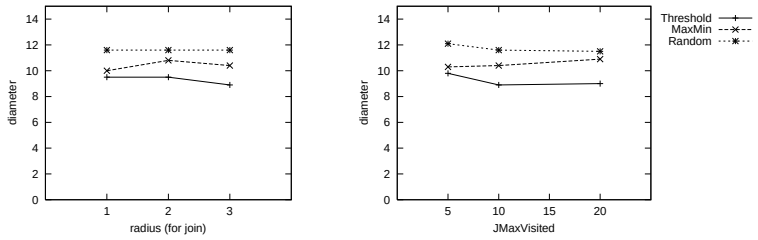

Fig. 3. Network diameter

Quality of Clustering We use two measures for evaluating clustering: join errors (the percentage of peers that fail to find an appropriate group to attach) and the average distance between peer groups.

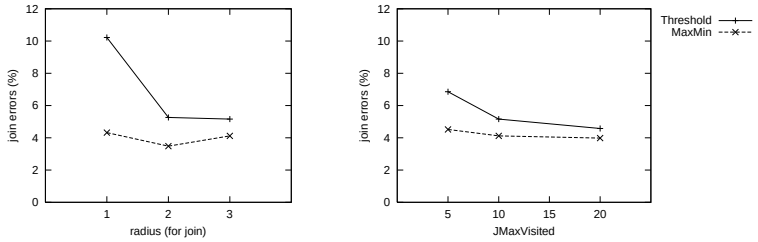

Fig. 4. Join errors

The percentage of peers that cannot find the appropriate group to attach affects the quality of grouping. The number of join errors decreases as the join radius increases (Fig. 4 left). This is because, with a larger join radius, the join message is propagated more efficiently, since each routing index summarizes the content of more peers in the network. The benefit of increasing the radius above 2 is negligible (or there is no benefit), which means that a small radius suffices.

Also, the number of join errors increases as the percentage of the peers visited during the join procedure increases (Fig. 4 right). Again, increasing this percentage over 10% gives a small benefit, and thus, a small number of visited peers is required for achieving a good grouping. With the *threshold* join algorithm, the new peer connects to the first SL peers it finds that have similarity larger than the threshold t, whereas with the $MaxMin$ algorithm, the join procedure continues until the maximum number of visited peers is reached. Thus, the $MaxMin$ algorithm achieves a better grouping of the peers.

We define the distribution of the group peers as the average distance between a peer v and the rest of the peers in the same group GR: $db(GR) = \frac{\sum_{v \in GR} \sum_{u \in GR, u \neq v} dist(u,v)}{(|GR|(|GR|-1)}$. Figure 5 depicts the average distance for the groups created using the two join procedures and a random network. The networks built using our join algorithms have a smaller value for the distribution of the group peers than the random network. We achieve a small distribution for the group peers despite the fact that a new peer connects to only one relevant peer through a short link. Comparing the $MaxMin$ and the *threshold* algorithms, we notice that for small values of radius (1 and 2), the $MaxMin$ algorithm outperforms the *threshold* algorithm. In all the other cases, the distribution value is nearly the same. Also variations of the radius and the percentage of peers visited during the join procedure have negligible influence to the distribution of the group peers for networks created by the $MaxMin$ algorithm (in contrast to the *threshold* algorithm). The reason is that even for small values of the radius and peers visited, the number of join errors for the $MaxMin$ algorithm remains very small (as shown in Fig. 4).

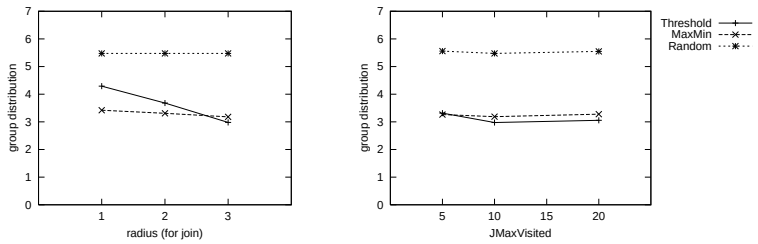

Fig. 5. Average distance between peers in each group

Comparison of the two join algorithms Figure 6 depicts the percentage of peers visited during the join procedure. With the *threshold* algorithm, the join procedure ends when the first SL relevant peers are found, whereas with the $MaxMin$ algorithm, it stops only when the maximum number of peers has been visited. Thus this percentage is 100% for the $MaxMin$ algorithm. Since, the *threshold* algorithm stops when it has found SL relevant peers to attach through short links (and a non relevant one through a long link), a smaller number of peers is visited. Especially, when the maximum allowed number of peers visited is 10% and above, the *threshold* algorithm takes a much smaller

number of hops during the join procedure. The drawback is that the *threshold* algorithm results in more join errors and a little worse performance (recall).

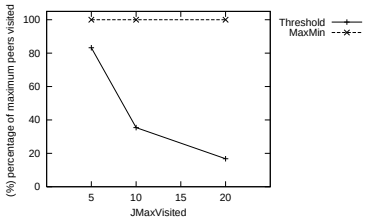

Fig. 6. Average percentage of the maximum number of visited peers during join

5.4 Performance of Querying

Recall The networks created by the two join algorithms perform better than the random network due to the grouping of the relevant peers (Fig. 7). The $MaxMin$ algorithm performs a little better than the *threshold* algorithm since it achieves better clustering. The performance is good for radius 1 and a small percentage of visited peers (5%), Increasing either the radius or the percentage results in improving performance. However, in most cases, increasing the radius over 2 and the percentage over 10% gives nearly constant performance on average.

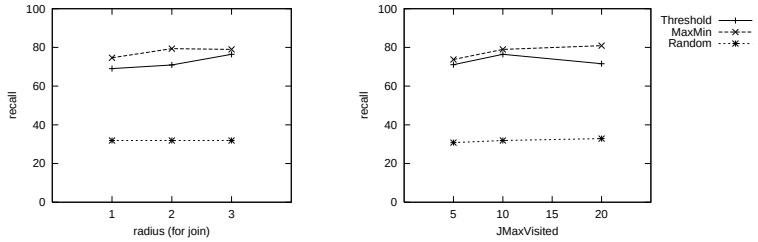

Fig. 7. Recall (percentage of matching peers found

Hops for First Result The average number of hops required to find the first matching peer is nearly the same for both the networks built using the *threshold* and the $MaxMin$ algorithm. Also, it is a bit smaller than the required number in a random network. Thus, only a small percentage of the network peers need to be visited to reach a matching peer. Then, it is easy to find most of the other matching peers since relevant peers are linked together.

6 Conclusions

In this paper, we have proposed building small worlds of peers based on the probability of peers matching the same set of queries. We have introduced a

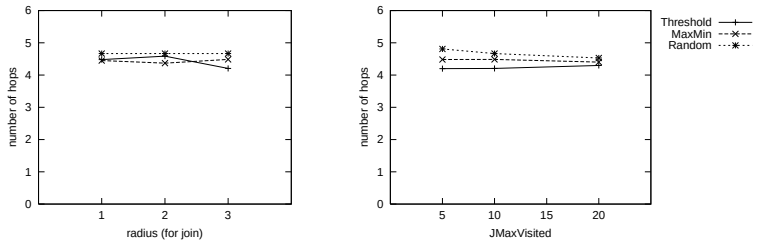

Fig. 8. Number of hops for locating the first matching peer

decentralized procedure for constructing such small worlds that is based on using routing indexes. Future work includes the definition of appropriate routing indexes that could work well to create small worlds and in addition take into account the query workload. We are also currently working on proving formally the properties of the networks constructed by our approach.

References

1. F. Banaei-Kashani and C. Shahaby. Searchable Querical Data Networks. In *International Workshop on Databases, Information Systems and Peer-to-Peer Computing*, 2003.
2. M. Bawa, G. S. Manku, and P. Raghavan. SETS: Search Enhanced by Topic Segmentation. In *SIGIR*, 2003.
3. B. Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. *CACM*, 13(7), 1970.
4. A. Crespo and H. Garcia-Molina. Routing Indices for Peer-to-Peer Systems. In *ICDCS*, 2002.
5. A. Crespo and H. Garcia-Molina. Semantic Overlay Networks for P2P Systems. Technical report, 2002. Submitted for publication.
6. R. Morris I. Stoica, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. *IEEE/ACM Trans. on Networking*, 11(1):17–32, 2003.
7. A. Iamnitchi, M. Ripeanu, and I. T. Foster. Locating Data in (Small-World?) Peer-to-Peer Scientific Collaborations. In *IPTPS*, 2002.
8. M.S. Khambatti, K.D. Ryu, and P. Dasgupta. Efficient Discovery of Implicitly Formed Peer-to-Peer Communities. *International Journal of Parallel and Distributed Systems and Networks*, 5(4):155–164, 2002.
9. G. Koloniari, Y. Petrakis, and E. Pitoura. Content-Based Overlay Networks for XML Peers Based on Multi-Level Bloom Filters. In *International Workshop on Databases, Information Systems and Peer-to-Peer Computing*, 2003.
10. A. Loser, F. Naumann, W. Siberski, W. Nejdl, and U. Thaden. Semantic Overlay Clusters within Super-Peer Networks. In *International Workshop on Databases, Information Systems and Peer-to-Peer Computing*, 2003.
11. S. Manku, M. Bawa, and P. Raghavan. Symphony: Distributed Hashing in a Small World. In *USENIX Symposium on Internet Technologies and Systems*, 2003.
12. D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu. Peer-to-Peer Computing. Technical Report HPL-2002-57, HP Laboratories Palo Alto, 2002.

13. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A Scalable Content-Addressable Network. In *SIGCOMM*, 2001.
14. S. C. Rhea and J. Kubiatowicz. Probabilistic Location and Routing. In *INFOCOM*, 2002.
15. C. Schmidt and M. Parashar. Flexible Information Discovery in Decentralized Distributed Systems. In *HPDC*, 2003.
16. C. Tang, Z. Xu, and S. Dwarkadas. Peer-to-Peer Information Retrieval Using Self-Organizing Semantic Overlay Networks. In *SIGCOMM*, 2003.
17. The MD5 Message-Digest Algorithm. Rfc1321.
18. P. Triantafillou, C. Xiruhaki, M. Koubarakis, and N. Ntarmos. Towards High Performance Peer-to-Peer Content and Resource Sharing Systems. In *CIDR*, 2003.
19. D. J. Watts and S. H. Strogatz. Collective Dynamics of Small-World Networks. *Nature*, 393:440–442, 1998.