Structuring Peer-to-Peer Networks Using Interest-Based Communities

Mujtaba Khambatti, Kyung Dong Ryu, and Partha Dasgupta

Arizona State University, Tempe AZ 85281, USA,
{mujtaba, kdryu, partha}@asu.edu

Abstract. Interest-based communities are a natural arrangement of distributed systems that prune the search space and allow for better dissemination of information to participating peers. In this paper, we introduce the notion of peer communities. Communities are like interest groups, modeled after human communities and can overlap. Our work focuses on providing efficient formation, discovery and management techniques that can be implemented to constantly changing community structures. We provide a mechanism to generate realistic peer-to-peer network topologies that can be used in simulations that evaluate the operation of our algorithms. Our experiments show how searching the peer-to-peer network can take advantage of peer communities to reduce the number of messages and improve the quality of search results.

1 Introduction

The current organization of the Internet allows users to connect to web servers of their interest that are often located using a search engine, such as [1–3]. In this paper, we propose an alternative organization built on an overlay network of peers. We provide a model for communication that scales well and efficiently uncovers community structures. We describe how these communities of peers can be used in structuring the peer-to-peer network.

A Peer-to-peer (P2P) system is a distributed system in which peers that have comparable roles and responsibilities, communicate information, share or consume services and resources between them [4]. These systems can harness massive amounts of storage with modest investment and no central authority [5, 6], and are therefore particularly attractive to everyday home computer users, who seem empowered by the ability to share a portion of the authority. The emergence of file-sharing applications, such as Gnutella [5], Freenet [6] and Napster [7], has been the catalyst that drew a lot of attention to P2P systems.

We introduce the notion of peer communities as a generalization of the multiplicity of peer groups (possibly overlapping) involving peers that are actively engaged in the sharing, communication and promotion of common interests. Our concept of peer communities is loosely based on the idea of "interest groups," for example Yahoo Groups [8] or Usenet Newsgroups, except that communities are self-organizing, and are formed implicitly due to the declared interests of human

users. We use communities as a more natural arrangement of distributed systems and show how they are helpful in pruning the search space. They also allow for better dissemination of information to participating peers, thereby reducing unnecessary communication within the P2P network.

Our solution for searching in P2P takes advantage of the self-organization of peers, and their capacity to form communities based on the interests of their human users and the interactions of individual peers. Earlier P2P search techniques, such as flooding, directed flooding, iterative deepening [9], and local indices [9], had the disadvantage that information located farther away from a peer could only be found through a considerable search expense. We believe that the community-based search query propagation provides more efficient searching by targeting one or more communities, irrespective of the current membership of the searching peer. Our technique follows the innate method of searching used by humans in the analogous social network, where queries for unknown items are asked to "those that know." The community-based search technique will allow search operations to be based on content rather than just filenames, as in many existing P2P search techniques [5, 6, 10, 11].

Efficient discovery and management of constantly changing community structures are essential to performing the proposed community-based search query propagation in a populated P2P space. In this paper, we show how these self-configuring communities are formed, discovered and managed, in spite of node failures. Finally, we discuss the mechanics of our community-based search solution and provide some initial evaluations of its performance. To demonstrate the performance of our algorithms, we use simulations to create populated P2P networks.

The paper is arranged as follows. Section 2 provides the motivation; Section 3 introduces some of the terms that we use; Section 4 describes how P2P networks are created; Section 5 illustrates our algorithms for structuring the network; and Section 6 explains the community-based search and provides simulation results. We conclude with Section 7.

2 Motivation

We view P2P networks consisting of an end-to-end overlay network of peers as being analogous to social networks comprising of humans. In fact, many of our proposed solutions for forming, discovering and managing structures in P2P networks, and their use to provide a more efficient search technique were motivated by our observations of similar solutions in social networks. For instance, the inclination of autonomous elements in a social system to form groups and associations leads us to believe that a populated P2P system made up of peers and an end-to-end overlay network, will also form similar community structures. Thus, P2P communities are a natural extension for arranging distributed P2P systems. Like their social network counterpart, these structures also enhance the capabilities of each member.

In this research we focused on understanding these community structures and proposing algorithms that can help manage and utilize P2P communities for better search operations, and consequently create newer P2P applications.

2.1 Forming and Discovering P2P Communities

A P2P community is a non-empty set of peers that share a non-empty set of interests that they have in common. Unlike a group, which is a physical collection of objects, a community is a set of active members, involved in sharing, communicating, and promoting a common interest. Peers in a network can exchange or advertise their interests with their neighbors in order to find peers with whom community relationships can be formed.

We proposed a community formation algorithm [12] that works without any central authority and optimizes the cost of communication. The algorithm is an autonomous procedure that is executed asynchronously by each peer. Through a simple exchange of interests, we demonstrated how communities could be formed. Each peer can then analyze the received interests to discover its community memberships without any additional communication.

2.2 Information Dissemination

P2P communities aid in the better dissemination of useful information amongst peers. For example, suppose node X belongs to a person interested in Amazonian Biological Catapults (ABC). After X declares this interest, it implicitly becomes a member of the community of ABC enthusiasts. Henceforth, all the information that X wants to share can be placed in a public directory that can be read/searched by all members of ABC. This concept can be extended to discover resources, physical devices, and network components. This example is interesting in the context of peer communities with no overlapping interests and is especially applicable in applications that follow the publish-subscribe-notify model [13].

2.3 Pruning the Search Space

Searching for information is one of the key challenges in P2P systems. Centralized searching has a drawback: the indexing and presentation of information are controlled by a central authority. P2P searching allows anyone to put up information in the search index and then cooperatively search the P2P space.

Consider a digital library built from a collection of peers, in which each peer owns a set of books that it is willing to share with other peers. The subjects of the books that a peer owns form its set of interests. Peers are implicitly grouped into communities based on the common interests they share. Since a peer could own books spanning a variety of subjects, it could be a member of multiple communities.

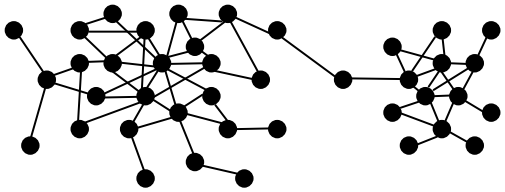


Fig. 1. Example of peer communities linked by a common peer. The vertices represent peers, and edges represent end-to-end connections between peers. The closely connected collection of peers to the left and the similar but smaller collection to the right are assumed to be two separate communities that are linked by a common peer

If the Computer Science (CS) and Medical (M) communities were disjoint, then search operations for medical information performed by a node that belonged to the CS community would not yeild any results. However, if the communities were linked at some point, lets say Q (Q belongs to both communities), then the medical information would be found, but at a great search expense, since, on the average, half of the CS community would be searched before a node from the M community is found.

To mitigate such problems, we need a community based query propagation method. Thus to provide efficient searching, it is better to search for one or more target communities, irrespective of the current membership of the searching node.

3 Terms and Definitions

In this section, we define and explain in detail two of the most commonly used terms in this paper: (i) Interest Attributes, and (ii) Peer Links.

3.1 Interest Attributes

Peer communities are formed based on their common interests. In our model, common interests are represented by attributes, which are used to determine the peer communities in which a particular peer can participate. Attributes can be either explicitly provided by a peer or implicitly discovered from past queries. However, there are privacy and security concerns in using such information, so we divide interests into three classes - *personal*, *claimed*, and *group*.

The full set of attributes for a peer is called *personal attributes*. However, for privacy and/or security reasons, all these attributes may not be used to determine community membership. A user may not want to reveal some of her personal attributes. Hence, a subset of these attributes is explicitly claimed public by a peer. We call these the *claimed attributes*. In addition, we introduce the notion of *group attributes*. Group attributes are location or affiliation-oriented and are needed to form a physical basis for communities. Every node belongs to

at least one pre-determined group and has a group attribute that identifies the node as a member of this group. The domain name of an Internet connection may be used as the group identifier.

The group attribute is also considered to be a personal attribute, which may or may not be one of the claimed attributes. It is recommended that a node include the group attribute as part of the claimed attribute set.

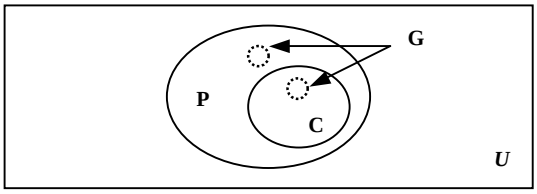


Fig. 2. Venn diagram of interest attribute sets for a peer. U is the universe of all attributes; P is the set of personal attributes; C is the set of claimed attributes; and G indicates the group attribute

The non-empty set of interest attributes that renders a collection of peers to become a community is called the *signature* of that P2P community.

3.2 Peer Links

P2P communities are attribute-based; that is, attributes (claimed and group) determine the membership of a node in one or more communities. In addition to attributes we also define an end-to-end overlay network in terms of *links*.

Links are not necessary to form and manage P2P communities. However, they are needed to feasibly run low-cost algorithms for formation and discovery, as it is conceptually and algorithmically simpler to use the notion of a set of *neighbors*[1] when communicating with other peers. To develop a better conceptualization of the nature of peer links, we explain a case where links are essential.

When node 'X' is born, it needs to have one or more logical neighbors. If it has three neighbors, 'A', 'B', and 'C', then we say that it has three links, X→A, X→B, and X→C. Unlike the overlay networks used by Distributed Hash-Table (DHT) based P2P systems, our link based network does not rely on node names, but on user selected neighbors. A peer, 'X' provides the following options that can be implemented to solve the isolation problem:

1. A special bootstrapping node that is present within each domain.
2. A peer that 'X' knows and trusts - a friend.

For a novice/new node, the first option may be the most appropriate. As 'X' ages, it finds other nodes and adds these links to improve the search speed and

[1] Neighbors are directly linked peers.

information access. The linkages are similar to friendships in real life, or to http links in the Web and are directed by humans.

Definition 1. *Link weights are given to each claimed attribute of a peer based on the percentage of links from the peer that can reach, after at most one indirection, other peers claiming the same attribute.*

Link weights are computed after the *escalation* of attributes (described in Section 5.1) and are important in determining the membership of a peer in a community.

4 Modeling a P2P Network

A P2P network can be thought of as a graph where the nodes represent peers and the edges represent the links between two peers. In this section, we explain two approaches for generating a realistic P2P network topology that could be used to simulate our algorithms.

4.1 The Internet as a P2P Network

The pre-cursor to the Internet (Arpanet) was one of the first P2P networks, which had the ability to connect computers as peers and provide them with equal rights in sending and receiving packets. Therefore, in order to evaluate our proposed algorithms, we initially evaluated them on an Arpanet successor - the Internet - as we know it today. Because it is difficult to make a large number of computers on the Internet run our programs, we wrote a spider program that would crawl a subset of the Internet and create a topological map on which we could simulate our algorithms. The spider program started at a user-specified website, requested its HTML content, and parsed the HTML code to extract all the linked websites. It recorded the websites that lay within a pre-specified domain, such as "asu.edu," and discarded the rest. Thereafter, the spider recursively visited each website from its recorded list, thus repeating the same steps. By programming the spider to travel the Internet domain using an Eulerian path, we could create a map of the web topology, where each node was a website and edges represented a link from one site to another.

Except for a few changes, the web topology graph that we generated closely resembled a P2P network since websites are analogous to peers and http links are manually created and analogous to peer links. One such change required converting the graph from a directed graph to an undirected graph where the edges between the nodes represented bi-directional links between peers. The other reason for this close resemblance was the website content, which was analogous to a peers interest attributes. Also, the domain specific attribute, such as "cse.asu.edu," is equivalent to the group attribute in a P2P network.

Although the web topology graphs demonstrated power-law properties (see Fig. 3), our calculations for small-world behavior identified a problem. In order

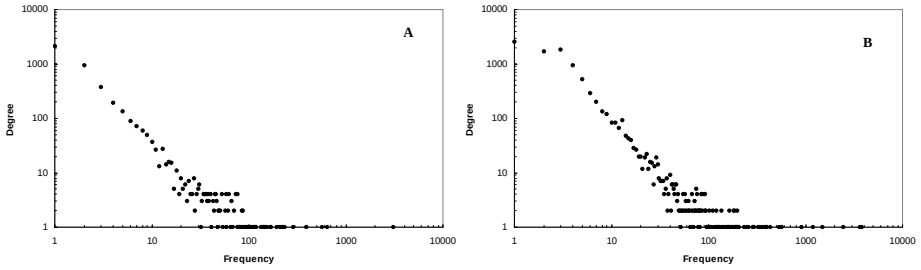


Fig. 3. The graphs show the power-law distribution of the frequency (*X-axis*) vs. degree (*Y-axis*) plot for the web topology graphs of: (*A*) "asu.edu" domain, and (*B*) "umich.edu" domain

for a graph or a network to be considered a small-world network, its characteristic path length must be as low as the path length in random graphs and its clustering coefficient (CC) must be as high as the CC in regular graphs. However, both these values were low in our topology graphs. The characteristic path lengths were 2.46 and 2.32 for asu.edu and umich.edu, respectively; and the clustering coefficients were 0.28 and 0.32 in the same order. We attribute this phenomenon to the popular use of increasingly efficient search engines on the Internet. While a few years ago, a website owner placed http links to frequently visited websites on her website so that they could be easily accessible, contemporary search engines efficiently locate websites so that many website owners do not even have to link to the websites of their colleagues and friends any more. The resulting topology graph therefore showed fewer regular links, and calculations for clustering coefficient revealed low values.

4.2 Creating Our Own P2P Network

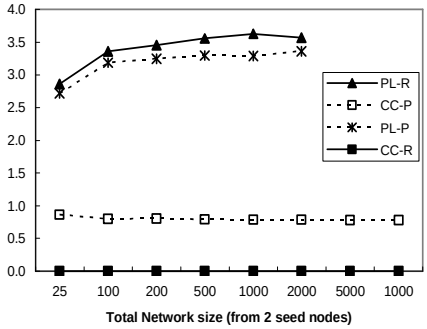
A P2P network comprising of peers that link to known peers is analogous to social networks, and therefore it should have a high clustering coefficient to represent the interconnected social links amongst circles of friends. We needed to provide a mechanism to ensure that our P2P network topology would exhibit the properties of a small-world network and would also show a power-law distribution for frequency versus degree.

Our next approach involved enforcing certain rules on new peers that wanted to join the P2P system. By virtue of these rules, the P2P system that was formed was a small-world network, which also exhibited a power-law (or scale-free) characteristic for the distribution of the number of neighbors of each peer.

We were inspired by the work of M. Steyvers and J. Tenenbaum [15] on semantic networks and extended the domain of their model to a P2P network that involves peers and links.

The new peer 'X' has to follow the rules below in order to join a P2P system:

1. Peer 'X' selects a single peer, 'A', from a list of known peers (see Section 3.2) that are currently members of the P2P system, such that 'A' is one of the



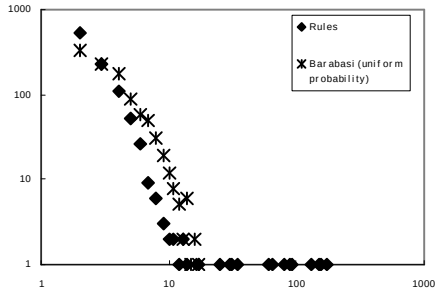


Fig. 4. (*left*) The graph shows the Clustering Coefficient (*CC*) and the characteristic Path Length (*PL*) for various network sizes generated with an initial seed of 2 nodes. The suffixes *P* and *R* indicate P2P and random networks, respectively

Fig. 5. (*right*) The graph shows the power-law distribution in a frequency (*X-axis*) Vs degree (*Y-axis*) plot of our rules mechanism compared with the well-known "Barabasi" technique. The network size was 1000 nodes - grown from 2 seed nodes

more well-connected peers, i.e., it has, on the average, more links to other peers within the P2P system than the other peers from the list.

2. Peer 'X' creates links to N neighbors of 'A', such that the neighbors of 'A' that have more links to other peers are chosen with a higher probability than the other neighbors.

5 Structuring the P2P Network

We have previously stated two important traits of P2P communities: they are implicitly formed, and their membership depends on the relationships between peers that share common interests. Below, we discuss in detail the techniques that we use to structure peers into communities.

5.1 Attribute Exchange, Escalation and Analysis

The exchange of interest attributes is not required in order to form communities. In fact, if all peers only tried to form communities based on their claimed attributes, we could significantly reduce the cost of communication. However, it is possible that a P2P community with signature 'S' might exist. As a result, a peer that has 'S' in its personal attribute set, but does not claim it, will not be able to join this community and avail of the benefits until it claimed 'S'.

Peers, therefore, need to expose (escalate from the personal list to the claimed list) as many attributes as possible in order to join the maximum number communities. This escalation can only be achieved by establishing communication with other peers.

After the escalation of attributes, a peer cannot assume to be a member of a community based on "unescalated claimed attributes". Only after analyzing the received interest attributes will a peer be able to discover the two kinds of communities: (i) the communities that peers are explicitly a part of, by virtue of their common group attribute or other claimed attributes; and (ii) the communities in which peers become members, by virtue of their claimed attribute set after escalations.

5.2 Discovering Community Members

Since interest attributes are constantly changing values, the attribute exchange process needs to occur on a regular basis to keep the P2P system up-to-date and the peers subscribed to the most suitable, existing communities. Then, again, a periodic increase in communication messages might not be suitable for low bandwidth networks, as regular communication will be affected by this increased traffic. Our solution is to opt for *Distributed Discovery* and *P2P Gossiping*. The nature of the Distributed Discovery algorithm enables a peer to become aware of the following information: (1) the approximate size of its communities, (2) the other community members, and (3) various profile information of the member peers, such as peer involvement values, and claimed interest attributes for each peer.

In addition to link weights, peers have *involvement* values associated with every community in which it is a member. Involvement is proportional to the number of peers in the *neighborhood*[2] that claim the signature attributes of a particular community. Therefore peers with a higher value of involvement associated with a community such as 'C', have more peers within their neighborhood that are also members of 'C'. We use the term *seers* when referring to these peers.

In [14] we have shown that information stored on the seers will be available to a large percentage of peers within the community. The Distributed Discovery algorithm described in the same paper was also shown to be a low overhead, simple protocol that was resilient to failures and delays in peers. The protocol used vectors traveling a Hamiltonian path, and it terminated easily. If random peers initiated this protocol within their communities, the end result would be a well-structured P2P network, with peers being aware of the configuration of their communities. Below we describe a variation of the distributed discovery protocol that is bound by a maximum hop-count for discovery in very large communities.

Hop-bound Distributed Discovery For communities that are extremely large, the Distributed Discovery algorithm will require a long time to conclude. Therefore the initiator will have to remain online for a long time to receive incoming vectors. To overcome this obstacle, we propose an alternative hop-bound Distributed Discovery that works by sending a maximum hop count (h) value along with the vector so that a sub-set of the community can be discovered. At

[2] The neighborhood of a peer includes neighboring peers and their neighbors.

a later stage, a merging algorithm can be executed for the purpose of combining various sub-sets into one community.

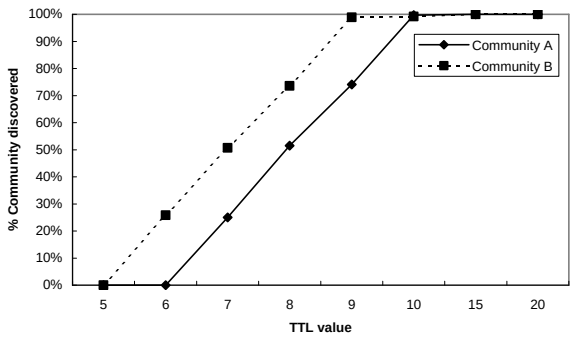


Fig. 6. The above graph shows the increasing percentage of the community discovered as the value of hop count increases. Note that the behavior of the hop-bound distributed discovery is linear. The test was conducted on a network with 4,500 nodes

The merging algorithm can be executed as a low priority activity, which is not essential to the operation of algorithms, such as community-based search. That being said, the merging algorithm helps structure the P2P network so that the search algorithm will work more efficiently. Fig. 7 below shows the cases that might occur during the hop-bound discovery of a community. Following Fig. 7 is the construct of the merging algorithm:

Case 1: *There exists more than one initiator within h hops.* If the initiators have neighbors within or beyond h hops, then by virtue of the Distributed Discovery algorithm, the vector with the lower identification number survives, and remains as the only initiator in the community until the process is terminated. The ousted initiator knows the identity of the extant initiator. All the results sent to the ousted initiator are forwarded to the extant initiator of the community.

Case 2: *There exists more than one initiator beyond h hops.* If the end vectors received by the initiator indicate that the hop count has been reached, then there could be some potential community members beyond h, who might have been involved in their own hop-based discovery. The initiator therefore sends a message to the peers that sent the end vectors requesting them to obtain the identity of the initiator from their neighbors that were not involved in this particular discovery. The initiator with the lowest peer identification value takes over as the new initiator of the merged community. However if no such initiator is found, then this operation of locating an initiator beyond h is repeated periodically so that eventually a merge operation will take place.

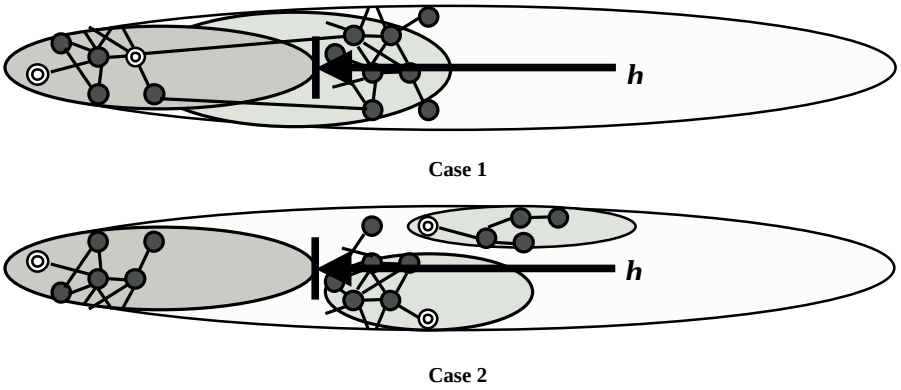


Fig. 7. The figures portray the two cases that might occur during a hop-bound distributed discovery. The large ellipse that encloses everything indicates the actual size of a community ($h = \infty$). The smaller ellipses indicate sub-sets of the community ($h \in \mathbb{N}$), where $\mathbb{N}$ is the set of natural numbers that are discovered. Peers (gray circles) and their links (connecting edges) are shown in the figure. Initiating peers are marked with concentric circles

Obtaining Bloom Filter Summaries Bloom filters [16] are compact data structures that probabilistically represent the elements of a set. They support queries of the form, "*Is X an element of the represented set?*" and at the current stage, they have been used in a variety of ways [17–20].

We extend our proposed Distributed Discovery protocol further by gathering Bloom filter summaries from each participating peer. The initiator creates a Bloom filter from its claimed attributes and sends it along with the vector. Each peer that receives the vector and the filter creates its own Bloom Filter and merges it into the existing filter. After the end vectors and filters are received, the initiator merges all the filters, forming, in this manner, a Bloom Filter that represents a compact summary of the attributes claimed by the peer members of a particular community.

In order to reduce the rate of false positives that result from the probabilistic nature of these data structures, we chose $k = 8$ hash functions and set the Bloom Filter size to $m = 2048$ bits. Based on formula (1) (described in [21], and [22]), we get: $p_{err} \approx 1.\mathsf{E}^{-05}$ for $n = 70$ possible claimed attributes.

$$p_{err} \approx \left(1 - e^{-\frac{kn}{m}}\right)^k \text{(false positives)} \tag{1}$$

The reasons for this modification are justified in the next section where we employ the Bloom Filter for a Community-Based Search algorithm.

Pseudo-Code for Constructing the Bloom Filter

```
Start-prog
```

```
bloomFilter = new bool[m] initialized to false
Foreach attribute in Claimed Attribute List
   Foreach hashFunction hi
      bloomFilter[hi(attribute)] = true
   End-for
End-for
End-prog
```

Pseudo-Code for Merging Bloom Filters

```
Start-prog
mergedFilter = new bool[m] initialized to false
Foreach bFilter in List of Bloom Filters to merge
   Foreach bElement in bFilter
      i = bloomFilter.IndexOf(boolElement)
      mergedFilter[i] = mergedFilter[i] bit-OR bElement
   End-for
End-for
End-prog
```

6 Community-Based Search

Our solution for searching P2P networks takes advantage of the self-organization of peers and their capacity to implicitly form communities. In this section, we briefly describe the mechanics of our search technique and provide some preliminary comparisons with known search algorithms.

6.1 Constructing the Search Query

Any peer that needs to search the P2P network constructs a three-part search query containing: (1) the identity of the peer creating the query, (2) the actual query for an item, and (3) a list of meta-information that describe the item. Meta-information descriptions are analogous to adding the word "television" after "Japanese" while doing an Internet search in order to narrow the search for Japanese televisions.

In an interest-based P2P network, such as the digital library from Section 2.3, a peer might use interest attributes as meta-information to a query. For instance, if the query is for "Vampires," the list of meta-information might include attributes such as, "Twentieth century," "Bram Stoker," and "European authors".

6.2 Processing the Query

To facilitate the search operation, the query is sent to the peer P_S that is most likely to either solve the query or know some peer that can solve it. In our digital

library example, a solution to a query could mean that the peer either owns the requested books or can provide information about the peer that owns them. As previously mentioned, this approach is markedly different from the traditional P2P search techniques. The peer P_S is chosen if it is a seer within the appropriate communities. At the end of Section 5, we described how Bloom filter summaries of claimed attributes were obtained for a particular community. At the current stage, the bloom filters are consulted to determine whether a peer claims any attribute from the meta-information list of the query. False positives can occur, thus increasing the overhead of locating P_S.

After the query is constructed, it is sent to the closest peer P_S that is also a seer in a community whose signature contains at least one attribute from the meta-information list. If the querying peer P_Q matches this description, then it is chosen to process the query. Else, P_Q looks for P_S from its immediate neighbors. In the case that P_S is not located, P_Q asks its neighbors to provide the identity of P_S. In the P2P networks generated using the rules described in Section 4.2, we found that the latter case occurs about 32% of the time. In addition, we found that P_S is almost always located after asking the neighbors. The cost involved in locating P_S is amortized over a number of queries because peers remember the identities of the closest peers that are seers of a particular community.

The query is then sent to P_S to be placed on the *blackboards* for the communities in which it is a seer and whose signatures contain at least one attribute from the query's meta-information list. Blackboards are similar to web pages and are independently maintained by a peer. Any peer can view the content on the blackboard of any other peer, provided that it knows the identity of that peer so that the blackboard can be reached. Peers maintain separate blackboards for each community in which they are members.

6.3 Checking Blackboards

Periodically and asynchronously, for each community C that it is a member, a peer visits the blackboard for C, which is maintained by each of its neighbors who are also members of C. If the visiting peer can solve any of the queries on the blackboard, then a message is created and sent to the peer that created the query. The message created could be sent via email to the querying peer.

Regardless of the outcome of the above procedure, the visiting peer copies the queries and places them onto its own blackboard for the community C. Our experiments revealed that even such an asynchronous, background communication amongst peers results in quick and efficient solutions to queries.

6.4 Simulation Results

We simulated the Community-Based Search (CBS) operation over a P2P network created by the rules described in Section 4.2. A set (10% of available peers) of random peers was selected to create queries. The details of the queries were randomly generated from a list of 25 known attributes. On an average, a query

had 3 attributes in the meta-information list. The performance of CBS was evaluated against the performance of two well-known search techniques: (1) Gnutella search, a hop-limited breadth-first search of the P2P network beginning from the querying peer; and (2) Hub search, a hop-limited search like Gnutella, except that only one peer, selected for having the maximum number of neighbors, is forwarded the query each time.

In the digital library, the participating peers created two kinds of queries for books: (1) queries containing the title of the book; and (2) queries containing partial book contents or genre descriptions. CBS can operate using either type of query. Our tests were performed using queries that are of type (2). Therefore a solution to a query contained a list of all the peers that matched as many genre descriptions as possible, implying that the peers were likely to be members of the communities $C = \{C_1, \ldots, C_n\}$ whose member/s owned the requested book.

The parameters with which the search techniques were evaluated are: (1) the number of messages required for each search method, and (2) the quality of the solution, which is a measure of the number of peers found that are likely to be members of the maximum communities in C, i.e. the peers found have high link weights for the attributes that matched the genre descriptions.

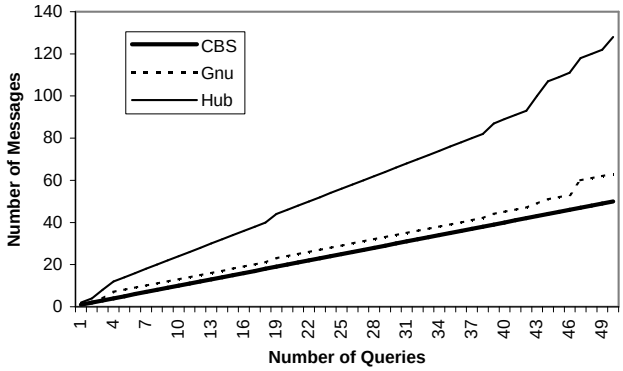


Fig. 8. Evaluation based on number of messages for search operation as the number of querying peers increases (*X-axis*). The results are the average values of 5 separate tests

Fig. 8 shows that CBS consistently requires fewer messages in order to process the queries. Furthermore, CBS scales well when the number of queries increases. On the other hand, the number of messages generated during the Hub-search method (*Hub*) begins to increase rapidly as more queries are created, because each hub forwards queries to all of its neighbors. Although the Gnutella (*Gnu*) technique exhibits linear behavior (since hop-limit was set at 5), it still does not out perform CBS in terms of number of messages generated.

Higher link weights (in fig. 9) for attribute 'A' indicate a higher probability that the peer is a member of a community of peers that claim attribute 'A'. For

most attribute values, CBS finds a peer with higher link weights than the other two techniques.

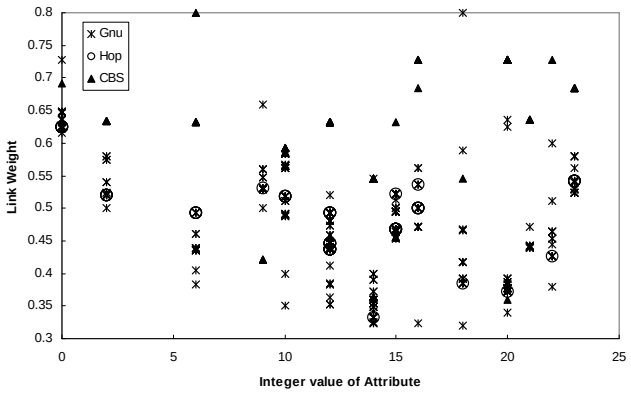


Fig. 9. The graph above displays the link weights of the peers found by the various search techniques. The *X-axis* of the graph corresponds to the attributes converted to an integer value ranging from 0 to 24

7 Conclusion

Peer-to-peer networks are autonomously created, self-organizing, decentralized systems that appeal to everyday home computer users. We have shown that these networks can be organized into interest-based communities using simple formation and discovery algorithms. We explained how a P2P network topology can be generated, and then illustrated our techniques for structuring the P2P network. Finally, we described the community-based search protocol that exploits this arrangement of the P2P network in order to provide better search operations. The results of our simulations showed that community-based search used fewer messages and found peers that were more likely to solve the query than two other well-known search techniques.

References

1. Google. http://www.google.com/
2. Yahoo. http://www.yahoo.com/
3. AltaVista. http://www.altavista.com/
4. Khambatti, M., Ryu, K., Dasgupta, P.: Efficient Discovery of Implicitly Formed Peer-to-Peer Communities. In: Mickle, M.H. (ed.): Intl. Jour. of Parallel and Distributed Systems and Networks, vol. **5**(4). ACTA Press, Calgary (2002) 155–164
5. Gnutella. http://www.gnutelliums.com/

6. Clarke, I., Sandberg, O., Wiley, B., Hong, T. W.: Freenet: A distributed anonymous information storage and retrieval system. In: Workshop on Design Issues in Anonymity and Unobservability. (Berkeley, CA, USA, 2000) 311–320
7. Napster. http://www.napster.com/
8. Yahoo Groups. http://groups.yahoo.com
9. Yang, B., Garcia-Molina, H.: Efficient Search in Peer-to-peer Networks. In: International Conf. On Distributed Computing Systems. Vienna, Austria, 2002
10. Stoica, I., Morris, R., Karger, D., Kaashoek, F., Balakrishnan, H.: Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In: Proc. ACM SIGCOMM, (2001) 149–160
11. Druschel, P., Rowstron, A.: Past: Persistent and anonymous storage in a peer-to-peer networking environment. In: Proc. the 8th IEEE Work. Hot Topics in Operating Systems (HotOS), (Germany, May 2001) 65–70
12. Khambatti, M., Ryu, K., Dasgupta, P.: Peer-to-Peer Communities: Formation and Discovery. In: 14th IASTED Intl. Conf. Parallel and Distributed Computing Systems (PDCS), (Cambridge, MA, USA, 2002) 497–504
13. Gummadi, R., Hohlt, B.: Efficient Implementation Of A Publish-Subscribe-Notify Model Using Highly-Concurrent B-Trees (unpublished). (2000)
14. Khambatti, M., Ryu, K., Dasgupta, P.: Push-Pull Gossiping for Information Sharing in Peer-to-Peer Communities. In: Intl Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA), (Las Vegas, NV, 2003)
15. Steyvers, M., Tenenbaum, J. B.: The large-scale structure of semantic networks: statistical analyses and a model of semantic growth (submitted). To: 25th Annual Meeting of the Cognitive Science Society (CogSci). (Boston, MA, 2003)
16. Bloom, B.: Space/time Trade-offs in Hash Coding with Allowable Errors. In: Communications of the ACM, **13**(7). (1970) 422–426
17. Gribble, S.D., Brewer, E.A., Hellerstein, J.M., Culler, D.: Scalable, Distributed Data Structures for Internet Service Construction. In: Proc. of the Fourth Symposium on Operating Systems Design and Implementation (OSDI 2000), (San Diego, CA, 2000)
18. Gribble, S.D., Welsh, M., Behren, R.v., Brewer, E.A., Culler, D., Borisov, N., Czerwinski, S., Gummadi, R., Hill, J., Joseph, A.D., Katz, R.H., Mao, Z., Ross, S., Zhao, B.: The Ninja Architecture for Robust Internet-Scale Systems and Services. In: Special Issue of Computer Networks on Pervasive Computing, **35**(4). (2001) 473–497
19. Hodes, T.D., Czerwinski, S.E., Zhao, B.Y., Joseph, A.D., Katz, R.H.: An Architecture for Secure Wide-Area Service Discovery. In: ACM Baltzer Wireless Networks: Selected papers from MobiCom 1999. (1999)
20. Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Weimer, W., Wells, C., Zhao, B.: OceanStore: An Architecture for Global-Scale Persistent Storage. In: Proc. of the Ninth Intl Conf. on Architectural Support for Prog. Lang. and Operating Systems (ASPLOS 2000), (Cambridge, MA, 2000)
21. Fan, L., Cao, P., Almeida, J., Broder, A.: Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol. In: Proc. of ACM SIGCOMM '98, (Vancouver, Canada, 1998)
22. Ripeanu, M., Iamnitchi, A.: Bloom Filters Short Tutorial (unpublished) (2001)