This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the ICC 2007 proceedings.

Speed Up Queries in Unstructured Peer-to-Peer Networks

Zhan Zhang Yong Tang Shigang Chen
Department of Computer & Information Science & Engineering
University of Florida, Gainesville, Florida 32611–6120
{zzhan, yt1, sgchen}@cise.ufl.edu

***Abstract*—Unstructured peer-to-peer networks have gained a lot of popularity due to their resilience to network dynamics. The core operation in such networks is to efficiently locate resources. However, existing query schemes, e.g., flooding, random walks and interest-based shortcut, suffer various problems in reducing communication overhead, and shortening response time. In this paper, we study the problems in prior works, and propose a new query scheme by mixing inter-cluster queries, and intra-cluster queries. Specifically, the proposed scheme works by efficiently locating the clusters sharing similar interests with inter-cluster queries, and then exhaustively searching the nodes in the found clusters with intra-cluster queries. To facilitate the scheme, we propose a clustering algorithm to cluster nodes that share similar interests, and a labeling algorithm to explicitly capture the clusters's borders. As demonstrated by extensive simulations, our new query scheme can improve the system performance significantly by delivering a better tradeoff between communication overhead and response time.**

I. Introduction

Peer to peer networks surged in popularity in recent years. The core operations in most peer-to-peer networks is to efficiently locate data items, in which the fundamental challenges are to achieve faster response time, smaller network diameter, and better resilience to network dynamics.

Structured P2P networks have been proposed by many researchers [1], [2], [3], [4], [5], [6], [7], in which distributed hash tables (DHT) are used to provide data location management in a strictly structured way. While structured P2P networks can offer better performance in response time and communication overhead for query procedures, they suffer from the large overhead for overlay maintenance due to network dynamics.

Unstructured P2P networks such as Gnutella rely on a random process, in which nodes are interconnected in a random manner. The randomness offers high resilience to the network dynamics. However, they rely on flooding for users' queries, which is expensive in computation and communication overhead. Consequently, scalability has always been a major weakness for unstructured networks [8].

Searching through random walks are proposed in [9], [10], [11], in which incoming queries are forwarded to the neighbors that are chosen randomly. In random walks, there is typically no preference for a query to visit the most possible nodes maintaining the queried data, resulting in long response time.

Interest-based shortcut [12] exploits the locality of interests among different nodes. The basic principle behind this approach is that a node tends to revisit accessed nodes again since it was interested in the data items from these nodes before. However, this approach may raise new problems, and it may even work worse than uniform random walks as discussed later.

In this paper, we take the unstructured approach, and propose a new query scheme to address these problems, which is able to deliver a better tradeoff among response time and communication overhead.

The rest of the paper is organized as follows. Section II reviews the possible problems in prior works, and gives the motivation of our scheme. Section III defines the interest similarity, and proposes a light-weight algorithm to cluster nodes within the same interest group and a labeling algorithm to explicitly border the clusters. Section IV introduces our query scheme in details. Section V evaluates the scheme with extensive simulations. Section VI draws the conclusion.

II. Problems and Motivation

A. Problems in Prior Works

Current query schemes suffer various problems in achieving a better tradeoff between communication overhead and response time.

Flooding: Flooding, e.g., [13], [14], [15], is a popular query scheme to search a data item in fully unstructured P2P networks, such as Gnutella. While flooding is simple and robust, its communication overhead, i.e., the message number, increases exponentially with the hop number, and most of these messages visit the node that have been searched. Consequently, communication overhead and scalability are always the main weakness in this approach [8], [16].

Random Walks: Random walks [9], [10], [11], [17] rely on query messages randomly selecting their next hops among neighbors to reduce the communication overhead, but this approach takes a long time to locate queried data items. If a network is well-clustered (nodes with similar interests are densely connected), the query latency is expected to be reduced significantly. However, it is not true, because the chance of a random walk message escaping out of the original cluster increases exponentially with the ratio r, defined as the inter-cluster edge number to the intra-cluster edge number. In the case of a network with a small value of r, e.g., $r < 0.01$, and the queried data items are in different clusters from the source node, a query message has to walk a long distance to be able to traverse the cluster border and locate the queried data items. In the case of a network with a large value of r, e.g., $r > 0.1$, and the queried data is in the original cluster, a query message may escape out of the original cluster within a small number of hops, resulting in a long response time. Consequently, random walks may suffer long response time regardless of the network having been well-clustered or not.

Interest-based shortcut: Interest-based shortcut, e.g.,[12], tries to avoid the blindness in random walks by favoring nodes sharing similar interests with the source, which can be regarded as a variation of markov random walks. However, it causes new problems. Suppose nodes in an interest group have formed a cluster, and query messages can be artificially confined in this specific cluster. In the sense of nodes in the cluster share similar interests, any of them possibly maintains the queried data. Thus, the query message should visit the nodes as many as possible, instead of visiting some specific nodes as soon as possible. However, due to the bias in selecting next hops in markov random walks, it tends to keep visiting some specific nodes, resulting in less distinct nodes being covered comparing to uniform random walks. Consequently, markov random walks work worse than uniform random walks if query messages can be confined in specific clusters.

B. Motivation

Researchers [12], [18], [19] have found many peer-to-peer networks exhibit small-world topology, and most of queried data items are offered by the nodes that share similar interest with the source node.

Intuitively, the nodes sharing similar interests with the source node should have higher priority to be searched than others. Practically, there are two challenges in designing such a query scheme. The first one is how to construct a small-world topology to cluster nodes sharing similar interests. By saying "similar interests", we actually mean that two nodes are interested with a common set of data items. Thus, the number of common accessed data items can serve as a metric to measure the interest similarity between two nodes. Based on this metric, we can design a clustering algorithm to densely connect the nodes with similar interests, and a labeling algorithm to enable a node to explicitly pick up its *inter-cluster* neighbors that have different interests from itself, and *intra-cluster* neighbors that share similar interests with itself.

The second challenge is how to fast locate the clusters that share similar interests with the source node, and how to exhaustively search nodes in the found clusters, if the queried data items are in the source node's interest group. This challenge is addressed by two types of queries: *inter-cluster* queries, and *intra-cluster* queries. The inter-cluster queries carry the interest information, and only travel on inter-cluster neighbors. The purpose of them is to fast locate the clusters that share similar interests with the source node. The intra-cluster queries are spawned by inter-cluster queries when a cluster sharing similar interest with the source node is hit. They only travel on intra-cluster neighbors for the purpose of exhaustively searching nodes in specific clusters.

Occasionally, queried data may be out of the source node's interest group, and possibly maintained by a cluster(s) with different interests. This problem can be addressed by blind search: inter-cluster messages randomly spawning intra-cluster messages when hitting clusters with different interests.

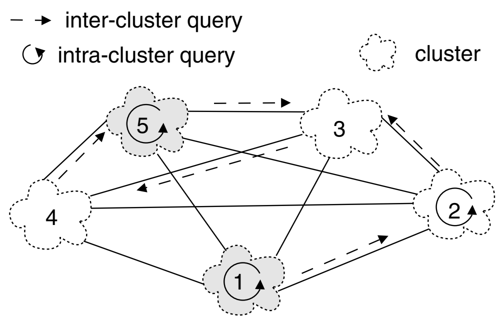

Fig. 1. A query scheme mixing inter-cluster queries and intra-cluster queries (the nodes in the grey clusters fall into the same interest group).

Take Figure 1 as an example. The network consists of 5 clusters, and nodes in the same cluster fall into the same interest group. Note that there exists an interest group consisting of two clusters: 1, and 5. In our schemes, inter-queries are initiated by a node in cluster 1, which travel among different clusters. By the interest information carried in the inter-queries, cluster 5 is found to share similar interests when it is hit, and an intra-cluster query is spawned, which then will exhaustively search the nodes in it. In addition, an intra-cluster query is spawned in cluster 2 by inter-cluster queries to support blind search.

III. Constructing a Small-world Topology

A. Measuring the Interest Similarity between Two Nodes

We start our discussion with the definition of interest similarity between two nodes in P2P networks.

If node u and node v share similar interests, then it is very likely that they have accessed same data items more or less previously. Therefore, the size of the common subset of accessed data items can serve as a metric to measure to what extent the interests of two nodes are similar.

However, a node may offer hundreds of data items, and hence, there may exist a large number of data items even in a small network. As a result, only if u and v have visited a large number of data items respectively, they are able to show some degree of similarity. An alternative to evaluate the interest similarity is by the number of common accessed nodes, which may enable a clustering algorithm to converge faster than the former approach. The problem in this approach is that two nodes visiting a common node does not indicate they have similar interests, because a node may offer data items belonging to multiple interest groups. For instance, a user u may offer resources for two groups: a set of mp3 music files for one group, and a set of research literatures for the other group. It is possible that two nodes visiting u have different interest. Thus, we have to address the discrepancy between the common set of accessed nodes and the common set of visited data items.

Suppose there are n nodes $N = \{1, 2, ..., n\}$ in the whole P2P network, and each node i offers a number of data items to the community. It categorizes (maps) all of these data items

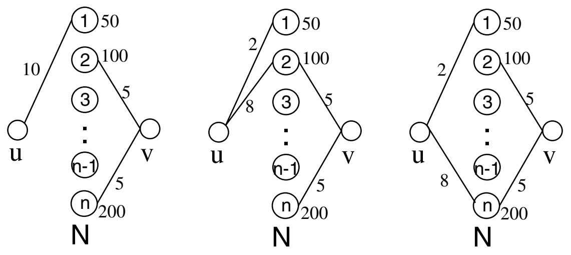

Fig. 2. Interest similarity between nodes. The number of data items in 1, 2, and n are 50, 100, and 200 respectively. left: no common visited nodes, center: u and v have visited node 2 (100 data items) with 8 and 5 times respectively, right: u and v have visited node n (200 data items) with 8 and 5 times respectively.

into α_i different categories, denoted as $C^i = \{c_1^i, c_2^i, ..., c_{\alpha_i}^i\}$. Suppose a data item x in i is mapped to a category $c^i(x)$, where $c^i(x) \in C^i$. How to categorize the data items is determined by the node i independently. For instance, node i may classify music files as category 1, while another node may classify music files as category 2. On the other hand, node i may fall into multiple interest groups, denoted as $G^i = \{g_1^i, g_2^i, ..., g_{\beta_i}^i\}$. For a node u, the access history with respect to each of its interest groups, e.g., g_1^u, can be specified by a set of data items x, denoted as $(i, c^i(x))$, where i represents the node offering the data item, and $c^i(x)$ is the category in C^i defined by i. If two nodes u and v share "similar interests", e.g., $g_1^u \approx g_2^v$, their histories for g_1^u and g_2^v tend to consist of a common set of $(i, c^i(x))$. Note that in the above definitions, each node determines its interest groups and categories independently without maintaining any any global information.

For easy explanation, we study a basic approach by assuming each user only falls into one interest group, and offers one category of data items. In this scenario, the access history can be represented by the accessed nodes alone. This approach can be easily extended to the multi-categories and multi-groups based on the definitions above.

The access history of a node u can be represented by a vector $V^u = (v_1^u, .., v_n^u)$, [1] where v_x^u represents how many times node u has visited node x. To cancel out the number of queries a node has issued, the access vector V^u is normalized to the *frequency vector* F^u, in which the i-th element in F^u is denoted as f_i^u, computed by V^u as $f_i^u = \frac{v_i^u}{\sum_{j \in N} v_j^u}$, representing the access frequency of the node i. Note that if a node i has not been visited by both u, then $f_i^u = 0$, indicating that a node does not need to maintain any information of unvisited nodes.

Moreover, given that two nodes u and v have visited a common node i, the probability of them visiting a common items correlates with the number of items offered by i, denoted as d_i. Take Figure 2 as an example, u and v in the middle part have more chance of having visited common data items than that in the right part because the number of data items in node 2 is half of that in node n. To account for this issue, we introduce a weighted diagonal matrix W with (i, i)-th value $w_{i,i}$ equal to $\frac{1}{d_i}$. It represents the probability of both u and v visiting a common data items, if both of them visit i once.

Now we define the following metric $A_w^{u,v}$ to evaluate the interest similarity between two nodes (Take the middle part in Figure 2 as an example):

$$
\begin{aligned}
A_w^{u,v} &= F^{u^T} W F^v \\
&= \begin{pmatrix} .2 \\ .8 \\ 0 \\ . \\ . \\ 0 \end{pmatrix}^T \begin{pmatrix} .02 & 0 & 0 & 0 & 0 & 0 \\ 0 & .01 & 0 & 0 & 0 & 0 \\ 0 & 0 & d_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & . & 0 & 0 \\ 0 & 0 & 0 & 0 & . & 0 \\ 0 & 0 & 0 & 0 & 0 & .005 \end{pmatrix} \begin{pmatrix} 0 \\ .5 \\ 0 \\ . \\ . \\ .5 \end{pmatrix} \\
&= 0.004
\end{aligned}
$$

Similarly, $A_w^{u,v}$ in the right part is equal to 0.002, indicating less interest similarity than that in the middle part.

If we view f_i^u and f_i^v as the probabilities of nodes u and v visiting node i, and $\frac{1}{d_i}$ as the probability of u and v visiting a common data items if both of them visit i, then the summation $A_w^{u,v}$ can be used to predict the probability that both u and v will visit a common data item in their future queries. Our definition is advantageous in manyfold. Due to space limitation, we omit the discussion in details.

B. Clustering Nodes with Similar Interests

Given the metric to evaluate the interest similarity between two nodes, we propose a light-weight clustering algorithm to connect nodes sharing similar interests.

In our strategy, each node i maintains a list L with limited size, e.g., 30, to record the nodes that possibly share same interests with itself. Each time a query message is processed, the similarity between the querying node itself and the node that owns the data items is computed. The newly obtained interest similarity, and the corresponding node's address are inserted into the list L. If the list is full, the stored neighbor with the lowest interest similarity is dropped.

By assuming that interests of nodes do not shift in a limited time frame, the nodes in L will serve as candidates of i's intra-cluster neighbors that sharing similar interests with itself.

C. Bounding Clusters

Although nodes with similar interests can be clustered by our clustering algorithm along with queries, existing query schemes, e.g., random walks, can only benefit marginally as discussed in Section II. To exploit the characteristics of the small world topology, our approach is to explicitly capture the clusters by each node i selecting inter-cluster neighbors in different interest groups, and intra-cluster neighbors in its own interest group.

A node i can learn its inter-cluster neighbors as follows. The node i issues a certain number of random walk messages only traveling on others' inter-cluster neighbors, and chooses the nodes hit by the messages as its inter-cluster neighbors. Note that inter-cluster neighbors should not overlap with the nodes in list L, which are candidates of i's intra-cluster neighbors.

[1]The real size of the vector V^u can be fixed to only record the nodes accessed most frequently by u

Thus, it is of the most importance to learn the intra-cluster neighbors. As discussed, for a node i, the nodes in L are candidates for its intra-cluster neighbors. We normalize the interest similarity of its neighbors j in L as:

$$p_{i,j} = \frac{A_w^{i,j}}{\sum_k A_w^{i,k}}$$

Naturally the transition probability $p_{i,j}$ can serve as a good metric to determine whether i and j are in the same interest group or not by introducing a threshold as a lower bound, denoted as T.

The purpose of intra-cluster neighbors is to confine intra-cluster queries within a specific interest group. Two nodes falsely regarded as intra-cluster neighbors may create a dramatic impact because an intra-cluster query may traverse to another cluster with different interests. On the contrast, two nodes i and k that are falsely regarded as inter-cluster neighbors will only have limited impact, because i and k may be connected by other intermediate intra-cluster neighbors j. In addition, the chance of i and k falling into the same cluster tends to become larger along with query procedures if they are in the same interest group. Based on this observation, T can be set as a relatively larger value. Suppose there are α neighbors in L that are possibly in the same interest group with i. In our paper, T is set to be $\frac{1}{\alpha}$. Note that $p_{i,j}$ and T are computed by node i locally, and the labeling algorithm does not involve any extra communication.

IV. Speed Up Queries

A. Query Messages

By explicitly capturing the cluster borders, we can formally define following three types of query messages.

The first one is called *l-query message*, denoted as M_l, which is a type of inter-cluster message only traveling on inter-cluster neighbors. The purpose of it is to quickly locate the clusters that may share similar interests with the source node and disperse intra-queries among different clusters.

The second one is called *s-query message*, denoted as M_s, which is a special type of intra-cluster message confining itself within a specific cluster by doing uniform random walks only on intra-cluster neighbors. An s-query message can only be spawned in a cluster sharing similar interests with the source node. The purpose of it is to exhaustively search nodes in the specific clusters. Each node having received the message should be able to estimate to what extent the cluster has been covered by the message. If most of nodes have been visited, an s-query message has little chance to discover more new nodes by keeping walking in the cluster, indicating the received message should be discarded. Otherwise, the message should be forwarded. Accurately estimating the covering time of a cluster is difficult and resource-consuming in a distributed system. Heuristically, if the message has been *consecutively* hitting a certain number, denoted as h, of nodes that have been visited by previous intra-cluster messages, the message is discarded. Thus, all messages in M_s need to maintain a counter to keep track on the number of consecutive nodes having been visited by previous intra-cluster messages.

The last one is called *b-query message*, denoted as M_b, which is also a special type of intra-cluster message similar to s-query message. The difference of b-query messages from s-query messages is that b-query messages may be spawned in clusters that have different interests from the source node. The purpose of it is to support blind search, because occasionally, the queried data may be out of the source node's interest group. The chance that the queried data item in a cluster having different interests is very small. Thus, once a b-query message hits a node that has been visited, the message is discarded to reduce the number of duplicated messages.

Moreover, to control the communication overhead, the total number of concurrent query messages has to be limited. It is achieved by the source node counting the number of l-query, s-query and b-query messages, denoted as m_l, m_s and m_b respectively. Whenever an intra-cluster message, e.g., s-query message or b-query message, is spawned or discarded, the corresponding counter in the source node needs to updated. Only if the summation of m_l, m_s, and m_b is smaller than a certain number, denoted as m, a new b-query message can be spawned to support blind search. In addition, all messages need to periodically check the status of the source node so that they can stop if the query has been successfully returned.

B. Mixing Inter-Cluster Queries and Intra-Cluster Queries

With three types of messages defined above, our query scheme is designed as follows.

Initialization: To initiate a query request, a node u issues a number m_l of l-query messages. If the queried data item falls in u's interest group, l-query messages carry the source's frequency vector, and a certain number m_s of s-query messages are issued to exhaustively search its own cluster. Otherwise, the message does not carry any interest information, and a b-query message is issued.

Receiving an l-query message: In the case of a node u receiving an l-query message, it calculates the interest similarity with the source node. If interest similarity is larger than a predefined value, a new s-query message is spawned and m_s in the source node is updated. Otherwise, a new b-query message is spawned, if the node has not been hit by other messages in M_s and M_b, and $m_l + m_s + m_b < m$. Finally, node u forwards the received message to a randomly selected inter-cluster neighbor.

Receiving an s-query message: In the case of a node u receiving an s-query message, if u has been hit by messages in M_s or M_b, it increases the counter in the message by 1. Otherwise, it resets the counter to 0. Next, if the counter is larger than the threshold h, e.g., 10, the node discards the message and notifies the source node to update the counter m_s. Otherwise, it forwards the message to a randomly selected intra-cluster neighbor.

Receiving a b-query message: In the case of a node u receiving a b-query message, if it has been hit by messages in M_s or M_b, the node discards the message and notifies the source node to update the counter m_b. Otherwise, it forwards the message to a randomly selected intra-cluster neighbor.

Our query scheme is *stateful*. If the same queries are reissued multiple times, less intra-cluster queries will be spawned in the well-searched clusters, and more intra-cluster queries will be spawned in the less-searched clusters, resulting in stronger ability to locate more replicas.

V. Simulation

In this section, the performance of the proposed clustering algorithm and query scheme is studied by simulations. If not explicitly defined, the number of nodes is $10,000$, and the average group size is 150. Each node maintains 1,000 data items, the average number of queries issued by each node is 30, the threshold h is equal to 10, and the probability of a node incorrectly classifying its queries or data items is 0.1. Moreover, m and m_l are set to be 32 and 16 respectively.

We compare our scheme to random walks, in which a source node issues 32 random walk messages in each query, correspondingly. In the figures, the legend "Uniform random walks (0)"/"Uniform random walks (1)" refers to the queried data items are out of/in the source node's interest group in the uniform random walks query scheme, and similarly "Inter-intra (0)"/"Inter-intra (1)" refers to the queries are out of/in the source node's interest group in the proposed scheme.

First, we study the effectiveness of our clustering and labelling algorithms. In Figure 3, it is observed that when the average query number is larger than 10, the algorithm reaches a stable state and almost all nodes in the same interest group form a single cluster. It indicates that our algorithm converge fast with the average number of queries, which is especially useful in highly dynamic P2P networks. By Figure 4, it can be observed that the average number of nodes in a cluster is almost the same as group size, demonstrating that $A_w^{u,v}$ can effectively measure the nodes' interest similarity.

Second we study the performance of our scheme with respect to query latency and communication overhead. In Figure 5, it is observed that if the queried data items fall into the original node's interest group, the number of hops needed for the majority of the queries is significantly reduced to about 20, while in the uniform random walks, it takes much longer time. Correspondingly, the number of messages is also much smaller in our scheme than that in random walks, as shown in Figure 6. The figures also show that if the queried data are out of the source node's interest group, the performance of our scheme is similar to uniform random walks.

We also have studied the performance of a network, in which each group consists of multiple different clusters. The results show the similar trends, which keeps true with respect to all other metrics that will be studied later. Moreover, the simulation results verify that the performance of random walks in a well-clustered network is similar to that in a poor-clustered network. Due to space limitation, we omit the details.

As have been observed, our scheme works much better than random walks in locating data items belonging to the same group as the source node. The reason behind it is that our scheme can discover more distinct nodes in the source nodes' interest groups within the same number of messages or hops, as shown in Figure 7 and Figure 8. It also indicates our scheme has stronger ability to locate more replicas, since it can discover a much larger number of nodes sharing similar interests.

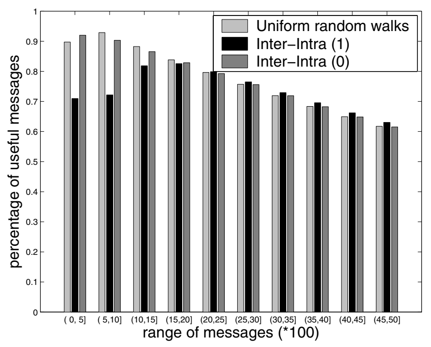

Fig. 9. The percentage of messages discovering distinct nodes within a certain message range.

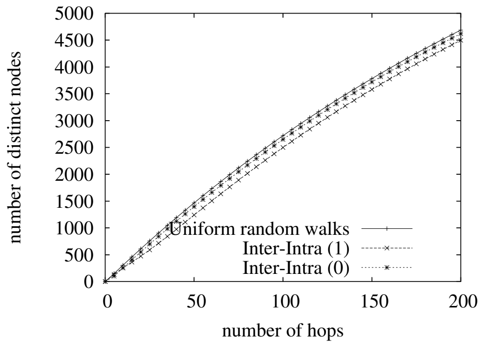

Fig. 10. The total number of distinct nodes discovered within a specific hop number.

Occasionally, the queried data item may be out of the source's interest group, or classified into wrong interest group by source node. In the former case, l-queries will not carry any interest information, but in the latter case, l-queries will carry wrong interest information. In both cases, the efficiency of our query scheme can be evaluated by the number of distinct nodes discovered by queries, including those out of the source node's interest group, within a certain number of messages and hops. Note that whether the queries are in or out of the original node's interest group makes no difference to random works. Figure 9 and Figure 10 show that in the first 1,000 messages, if the queries carrying interest information, less distinct nodes can be searched in our scheme. The reason is that s-query messages mistakenly exhaustively search the nodes in the clusters that share "similar" interests in the beginning, which has been demonstrated by our previous simulations. Consequently, the number of b-query messages is limited. Along with the increment of the number of messages/hops, our scheme works similar to the uniform random walks. It is because after most of nodes sharing similar interests are covered, more b-query messages will be spawned to search clusters with different interests, which are able to discover more distinct nodes. In addition, by the figures, if the queries carry no interest information, our scheme works similar to uniform random walks.

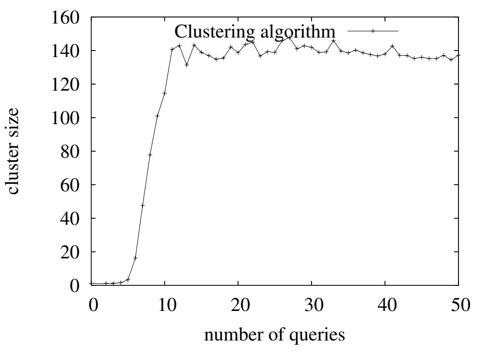

Fig. 3. The effect of average query number on the cluster size.

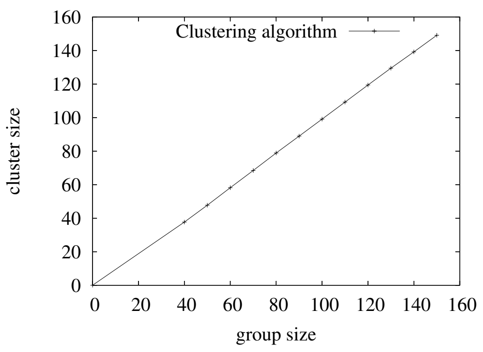

Fig. 4. The interest association is a good metric to estimate interest similarity

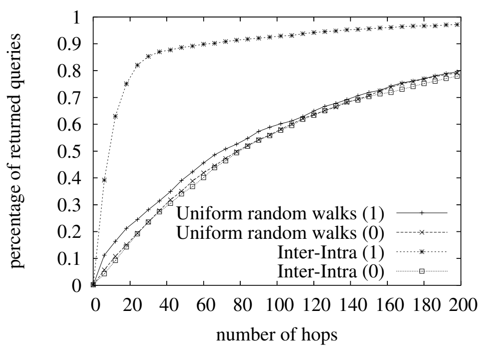

Fig. 5. The percentage of returned query within a specific hop number.

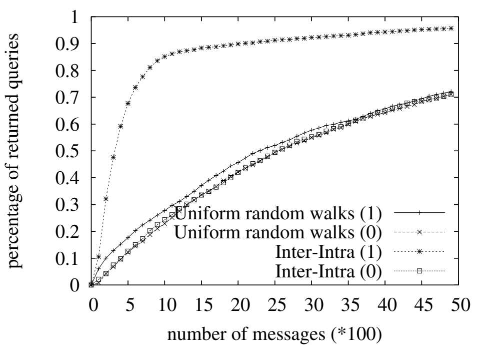

Fig. 6. The percentage of returned query within a specific message number.

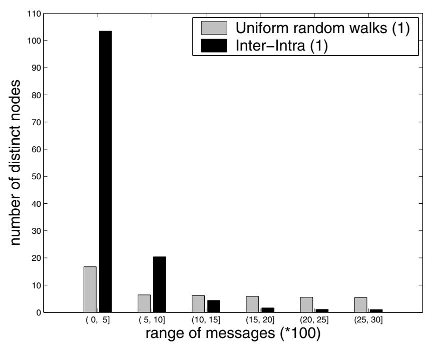

Fig. 7. The number of distinct nodes discovered in the same group within a certain message range.

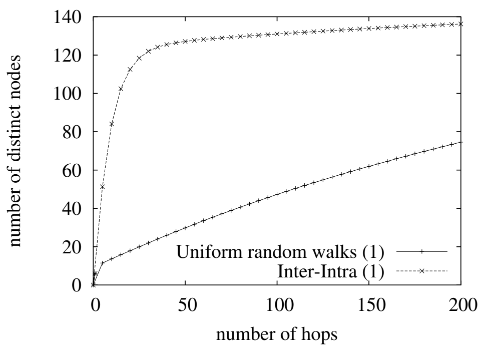

Fig. 8. The number of distinct nodes discovered in the same group within a specific hop number.

VI. Conclusion

In this paper, a clustering algorithm and a labelling algorithm are proposed to connect nodes sharing similar interests and capture clusters in the underlying networks explicitly. Then a query scheme mixing inter-cluster and intra-cluster queries is designed to speed up queries. Our scheme can achieve a better tradeoff among communication overhead, response time, and the ability to locate more resources (replicas). The performance of the algorithms has been demonstrated by simulations.

References

[1] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, F. M. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Transactions on Networking (TON)*, 2003.

[2] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A scalable content-addressable network," in *Proc. of ACM SIGCOMM'2001*. ACM Press, 2001.

[3] B. Y. Zhao, L. Huang, S. C. Rhea, J. Stribling, A. D. Joseph, and J. D. Kubiatowicz, "Tapestry: A global-scale overlay for rapid service deployment," *IEEE J-SAC*, January 2004.

[4] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Proc. of Middleware'2001*, Nov. 2001.

[5] C. G. Plaxton, R. Rajaraman, and A. W. Richa, "Accessing nearby copies of replicated objects in a distributed environment," *Theory of Computing Systems*, vol. 32, no. 3, pp. 241–280, 1999.

[6] D. Malkhi, M. Naor, and D. Ratajczak, "Viceroy: a scalable and dynamic emulation of the butterfly," *Proc. of ACM PODC'2002*, 2002.

[7] A. Kumar, S. Merugu, J. Xu, and X. Yu, "Ulysses: A robust, low-diameter, low-latency peer-to-peer network," in *IProc. of EEE ICNP'2003*, Nov. 2003.

[8] J. Ritter. (2001) Why gnutella can't scale. no, really. gnutella.html. [Online]. Available: http://www.darkridge.com/ jpr5/doc/

[9] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making gnutella-like p2p systems scalable," in *Proc. of ACM SIGCOMM'03*. ACM Press, 2003, pp. 407–418.

[10] C. Gkantsidis, M. Mihail, and A. Saberi, "Random walks in peer-to-peer networks," in *Proc. of IEEE INFOCOM'04*, Mar. 2004.

[11] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and replication in unstructured peer-to-peer networks," in *Proc. of ICS'02: the 16th International Conference on Supercomputing*. ACM Press, Sep. 2002, pp. 84–95.

[12] K. Sripanidkulchai, B. Maggs, and H. Zhang, "Efficient content location using interest-based locality in peer-to-peer systems," in *Proc. of IEEE INFOCOM'03*, Mar. 2003.

[13] N. Chang and M. Liu, "Revisiting the ttl-based controlled flooding search: Optimality and randomization." *Proc. of ACM MobiCom'04*, 2004.

[14] ——, "Controlled flooding search in a large network." *Proc. of Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2005.

[15] Y. Tang, Z. Zhang, S. Chen, and G. Fan, "A distributed hybrid scheme for unstructured peer-to-peer networks." *Proc. of ICC'06*, 2006.

[16] K. Sripanidkulchai. (2001, Feb.) The popularity of gnutella queries and its implications on scalability. gnutella.html. [Online]. Available: http://www.cs.cmu.edu/ kunwadee/research/p2p/

[17] C. Gkantsidis, M. Mihail, and A. Saberi, "Hybrid search schemes for unstructured peer-to-peer networks," *Proc. of IEEE INFOCOM'05.*, 2005.

[18] A. Iamnitchi, M. Ripeanu, and I. Foster, "Locating data in (small-world?) peer-to-peer scientific collaborations." *Proc. of IPTPS'02: 1st International Workshop on Peer-to-Peer Systems*, 2002.

[19] ——, "Small-world filesharing communities," *Proc. of IEEE INFOCOM'04.*, 2004.