A Dynamically Formed Hierarchical Agent Organization for a Distributed Content Sharing System

Haizheng Zhang Victor Lesser
Department of Computer Science
140 Governors Drive
Amherst, MA, 01003
{hzhang,lesser}@cs.umass.edu

ABSTRACT

The organization and collaborative protocols of agent societies are becoming increasingly important with the growing size of agent networks. Particularly, in a multi-agent based content sharing system, a flat, peer-to-peer(P2P) agent organization is not the most efficient organization for locating relevant agents for queries. This paper develops and analyzes a hierarchical agent group formation protocol to build a hybrid organization for large-scale content sharing system as well as a content-aware distributed search algorithm to take advantage of such an organization. During the organization formation process, the agents manage their agent-view structures to form a hierarchical topology in an incremental fashion. The algorithm aims to place those agents with similar content in the same group. We evaluate the system performance based on TREC VLC 921 datasets. The results of the experiment demonstrate a significant increase in the cumulative recall ratio(CRR) measure compared to the flat agent organization and structure.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: ,Distributed Artificial Intelligence, Multi-agent Systems

General Terms

Algorithms,Design,Experimentation

Keywords

Peer to Peer Networks, Agent Organization, Distributed Information Retrieval

1. INTRODUCTION

This paper investigates the role of an agent organization in a large-scale content retrieval system. In such a system, each agent shares its document collection and cooperates with other agents to conduct information retrieval tasks. An information retrieval task is defined as a process during which an agent receives a query from a user, forwards the query to other agents, who then conduct local searches on their own document collections and return relevant documents to the query initiator. In this paper, we consider such a system as a multi-agent system and focus on the organizational perspectives, i.e. how to organize these agents together to provide better system performance.

Our previous results showed that an unstructured search strategy performs badly on flat P2P agent networks. However, a topology reorganization process combined with context-aware search algorithms can improve the information retrieval performance considerably [9]. This has motivated us to investigate more complicated multi-agent organizations. Here we propose a hierarchical agent organization formation protocol to explicitly form a multi-level topical hierarchical structure to facilitate locating relevant documents. In this approach, the agents join different groups in the hierarchy based largely on their content similarity.

We make the following assumptions in this paper. First, each agent maintains an independent index and an IR search engine for its local document collection. However, we do not introduce any further restrictions on the local search engines and thus the network can be populated by agents having very different local search engines. Second, the experimental results presented are based on local search engines that are "perfect" in that they return all relevant documents in the collection for a given query. Third, we assume there is a third-party protocol in place to merge the returned results. Thus, our protocol does not have to deal with the merging of the returned lists. Lastly, we assume that agents are cooperative in that they all agree to use the same protocols for propagating resource descriptions among each other, accepting queries from peers and finally returning search results to the originators of the queries

The main contributions of this paper are as follows: (1) A group formation protocol for forming a hierarchical topical organization. The group formation is achieved by organizing the agent-view structures properly so as to place semantically similar agents together to form explicit groups in an incremental and distributed manner. (2) A content-aware search algorithm taking full advantage of the hierarchical organization. During the search process, agents in the network follow various cooperation strategies to forward queries and return results in the network.

The remainder of the paper is structured as follows: Section 2 presents the agent's internal structure and overall system architecture. Section 3 describes the algorithm to incrementally form the hierarchical organization of agents. Section 4 illustrates the content-aware distributed search algorithm. Section 5 presents the experimental framework used to evaluate search performance and analyzed the results of the experiments. Section 6 presents the related work. Section 7 concludes the paper.

2. THE SYSTEM ARCHITECTURE

In the proposed hierarchical agent society, agents have two roles: group-mediator and query-processor. All non-leaf agents in the organization take on both roles while leaf agents only take on the role of query-processor. Each mediator manages a group of agents and takes on a central role in group management including decisions on whether a new agent should be added to the group, when to reorganize the group, the selection of group members to handle a query and the propagation of queries to non-group members. In this section, we first describe the internal structure of an agent(Section 2.1) and then in Section 2.2, we introduce techniques for how agents can form a hierarchical organization based on topics.

2.1 The internal structure of an agent

Fig. 1 illustrates the internal structure of an agent. Each agent is composed of four components: the collection information, a local search engine, an agent-view structure and a control unit. The collection information includes the collection hosted by the agent to share with other agents as well as a collection model built for the collection. The collection model can be considered as the "signature" of a collection; a collection model is a statistical language model built for a particular collection. It characterizes the distribution of the vocabulary in the collection and estimates the probability of absent words using various smoothing techniques. The language model concept was originally introduced in information retrieval research [8] and has proven effective in the distributed IR applications [1], [9].

The language model has many interesting properties which are easily exploitable in peer-to-peer network systems: first, a collection model is lightweight as it significantly condenses the description of the content of the collection and thus is much smaller in size compared to the collection. Additionally, the size of the collection model grows minimally with the size of the document collection. Secondly, the collection model is a relatively accurate indicator for the content of the collection.

The agent control unit's role is to accept user queries, decide whether the queries should be processed by one or more group members, and determine the order of the other agents or groups that the queries should be forwarded to. The local search engine allows each agent to conduct a local search on its document collection so as to determine whether there are any documents that meet the criteria of a specific user query and then return relevant documents.

The agent-view structure, also called the local view of each agent, contains information about the existence and structure of other agents in the network and thus defines the

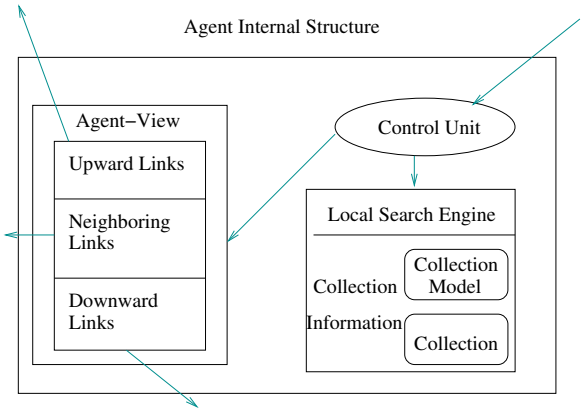

Figure 1: The internal structure of an agent

underlying topology of the agent society. The hierarchical group formation process organizes the agent-view structures of the agents to form a nearly-decomposable hierarchical structure. The agent-view structure also contains the collection model of the collections besides the address of these agents. The agent-view structure differentiates agents that take on both roles from those that take on only the role of query-processor. The agents who only take on the role of query-processor have only two kinds of links in their agent-view: neighboring links which connect neighboring agents together and upward links, which connect a member to its immediate and higher level group mediators. Besides these two kinds of links, mediators have downward links which connect a mediator to its immediate group members. In our current model, besides their direct mediators, agents also connect to the top-level mediator of their group with an upward link so as to facilitate the incremental group formation process as well as the search process, which will be introduced in later sections. This top-level connection is not an absolute necessity since there are many alternative linking structures that can achieve this goal. However, for simplicity in describing the protocol and in our implementation, we assume the existence of this type of link.

It is worth to pointing out that the degree, or the number of the entries in agent-view structures is an important factor in determining system performance. On one extreme, if each agent has the information about all agents in the network, it is indeed a centralized version of distributed information retrieval problem which has been well studied in the information retrieval community[1]. This, however, is impracticable in P2P system for a very large network. Previous studies indicate that in general the degree of each node satisfies the power law distribution statistics. In our work, we specify that each agent has a degree limit for neighboring links based on power law distribution assumption. Further we assume there is an underlying linear relation between the neighboring links degree and upward, downward links degrees. In this paper, we use a *degree ratio*, which is defined as the ratio of the upward(downward) links degree and the neighboring links degree, to capture this correlation.

2.2 The topic based hierarchical structure

In this paper, we attempt to create a topic based hierarchical structure on a multi-agent based content sharing network. In this structure, we define a content group as a set of agents including a mediator and all the group members which connect to the mediator directly or indirectly through upward links. We call a content group led by a top-level mediator a top-level content group. Inside each content group, all the agents store collections on a same topic. An agent can belong to multiple groups if applicable. In our nearly-decomposable hierarchical organization, the neighboring links connect agents in a peer-to-peer fashion while downward and upward links connect agents in a hierarchical fashion. These hybrid links make the system well connected, more load-balanced, and meanwhile have clusters of agents that are organized by topical structures.

Figure 2 illustrates an example of a hierarchical multi-agent based content sharing network. The solid lines between agents represent downward links and upward links. In this example, we simply assume upward links and downward links are symmetric, thus we use a solid line represent both links. The dashed lines between agents are neighboring links. The ellipse with solid line represents the top-level mediator network. In this figure, there are three topic-level mediators which lead content groups on "Sports", "Business" and "College" respectively. Through downward links, top-level mediator leading "Sports" group is connected to three immediate group members "Basketball", "Professional League" and "College Sports" respectively. Notice that the two agents "Basketball" and "Professional League" share a same group member which hosts NBA game information. We also stipulate that agents in different level can connect to each other via neighboring links. For example, "Basketball" is connected to "Business" through a neighboring link. This specification makes the "level" concept somewhat fuzzy in our system.

The potential advantages of such a structure are: (1) It allows for rapid locating a certain topic during the new agent joining process and search process. (2) During the search process, only a small subset of agents where most of the matching documents reside needs to be probed, thereby reducing the number of messages propagated in the network and the number of local document search. This can significantly improve the throughput of the system.

3. HIERARCHICAL GROUP FORMATION

In this section, we propose an online hierarchical grouping protocol. The protocol aims to build a topical hierarchical structure incrementally as agents join the network. The joining process for a new agent involves locating one or more appropriate groups (or starting a new group), deciding which group(s) to join and then merging itself into the desired group(s). Similar to the classical clustering problem, a natural question we have to answer is "how many top-level mediators should we have". This problem is not trivial even if we have global knowledge. Various solutions have been proposed to address this problem. In this work, we exploit a threshold based approach: we assume that each agent is associated with a threshold, called T_{GROUP}. If the likelihood of another agent belongs to a top-level content group is above the threshold T_{GROUP} associated with the top-level mediator, this agent can potentially be considered as a member of this top-level content group.However, if there is no top-level mediator whose collection model is close enough to the new agent, the new agent will start a top-level content group on it own.

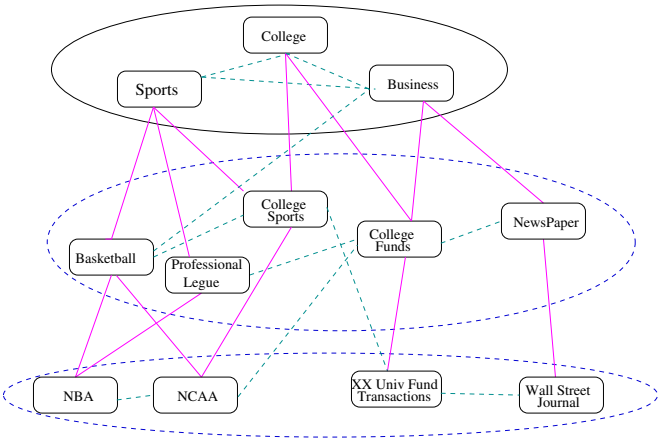

Figure 2: The system architecture

In the heteregeous environment, the topical similarity threshold T_{GROUP} can vary considerably for different topics. We therefore stipulate this threshold as an application-dependent value instead of trying to determine a value for all the situations. Due to the limited space, we do not present here how to calculate content similarity and determine topical threshold[10].

After locating a top-level content group, the protocol proceeds to locate the most appropriate group for the new agent by recursively splitting and inserting operations on the agent organization. Through these operations, agents always keep those agents with the most similar content as their immediate group members. During this process, the new agent builds its agent-view structure to place itself appropriately in the network. Meanwhile, the current existing members might prune their agent-view structure to reflect the changes and connect themselves to the new agent through neighboring links. Section 3.1 introduces the message propagation and group locating process; Section 3.2 describes the actual join procedure. To further clarify this procedure, [10] provides many examples and figures.

3.1 Message propagation and group locating

When a new agent joins the system, it first contacts any agent in the system with a *Join?* message. (Note that our protocol does not specify how to acquire the entry point information of the network. However, we assume the network is accessible either by an out-of-band approach as Gnutella does or by other mechanisms). The results of these messages are groups invitations from which the new agent chooses one or more invitations to join a specific part of the agent society. This process is carried out in a top-down fashion. Specifically, the new agent always tries to locate the appropriate top-level mediators whose similarity with the new agent is above their group threshold, T_{GROUP}. The top-level mediators will then either add the new agent as its own direct lower level member, or add it to a lower level group by pass-

ing the new agent to other members in the group. The rest of this section details the group locating process.

The *Join?* message includes the collection model and other information about a new agent. Upon receiving a *Join?* message from a new agent, say N_A, an agent forwards the message to its top-level mediator, say M, through the upward link described in the previous section. M compares the similarity of agent N_A's collection model and its group model, which is approximated by M's collection model, i.e $P(N_A|M)$. If the similarity is above the threshold associated with M, (i.e T_{GROUP}), M will start a procedure to generate an invitation for agent N_A to join the group it manages. Otherwise, M deems that the new agent is not close enough to be a member of the group. In either case, M forwards the *Join?* message to its neighbors(other top mediators) with a certain TTL(Time to Live) value. Each mediator who receives the *Join?* message repeats the same procedure as M does. The TTL value decreases by 1 when the message goes through each agent in the agent society. During this process, the new agent may join multiple groups if its collection model is close enough to those groups' group models.

There are two different procedures that can be followed in this action. In the first case, if the number of the downward links for the top-level mediator is below a pre-specified limit, the top-level mediator will simply invite the new agent to join its group. In the second case, if the downward degree of the mediator has reached a pre-specified limit, the mediator then starts to merge two of its group members into one subgroup in order to integrate the new agent. Note that the mediator makes the connection changes permanently regardless of whether the new agent accepts the invitation or not. Because the outcome of such a merging operation always makes similar agents more closer by moving one agent to a deeper level, we believe that the topology can be better off with such an operation. Furthermore, by specifying it this way, we avoid introducing transaction-like operations which would make the protocol considerably complicated.

3.2 Member Joining and mediator re-election

After sending out the *Join?* message, the new agent starts a timer. When the timer expires, the new agent examines the group invitations received from current agents in the network and decides which group(s) to join.

If the process ends up with no group invitations when the timer expires, or *NotAccept* messages from all the groups it wanted to join in, the new agent will start a new group with a threshold $P_e(D|G)$ which is pre-acquired through offline computation. It then becomes a top-level mediator in the network. If it receives multiple group invitations, we specify that the new agent will join all the groups as long as the number of its upward links is below the upward links degree limit. If the limit is reached, it picks the groups with the highest similarity to join. At this point, the new agent would always connect to the top-level mediator with an upward link.

Note that we set the mediator as the content centroid for the group. With more and more nodes joining in, the content centroid changes over time. For this reason, the current mediator periodically checks its group members and determines if it should hand over the mediator position to a new mediator. If this happens, the mediator will pass to the new mediator its mediator neighbors and all the information about the regular agents in the group to the new mediator. The structure of the group might change correspondingly as the degree of the old mediator and the new mediators could be different. This update will be sent to the affected agents to update their agent-view structures to reflect the recent changes.

Once the new agent and its group mediators update their agent-view structure, the new agent is now part of the agent organization. The new agent will then broadcast *Arrival* messages in order to build intra-level connections to other agents in the group. This process results in building neighboring links for both new agents and the currently existing agents.

Specifically, the new agent sends out *NewArrival!* messages with a certain TTL value to the mediators who then forward the message to their neighbors and or upper level mediators. TTL value decreases after each hop. During the message propagation process, each existing agent in the network also takes this opportunity to prune its neighboring list. Upon receiving the *NewArrival!* message, a current agent chooses to build a connection with the new agent and send out a *LinkInvitation* message with a certain likelihood. Once it decides to build a link to the new agent, the current agent simply adds the new agent as a neighbor if the current degree is below the capacity limit and otherwise, they will cancel one of the current links to take on the new agent. Upon receiving the *LinkInvitation* message, the new agent chooses which agent to connect to randomly. For the sake of simplicity, we specify that the issuing and acceptance of an invitation message are two independent processes, which implies that the current existing agent makes its own decision to issue *LinkInvitation* and cancel the current links even though the new agent may not accept the "Invitation" .

4. A TWO-STAGE SEARCH ALGORITHM

Search in a content sharing multi-agent system should avoid significant communication costs and minimize the access to those agents that do not contain relevant documents for a given query. The benefits of communication cost savings are obvious. However, avoiding unnecessary query processing that involves local searches of documents collection can also significantly improve overall system performance especially in situations where there are many active queries concurrently being processed by the system.

Our query search process is structured into two stages. In the first stage, a coordinated search protocol is used to relocate the queries to "relevant agent zones" by taking advantage of the hierarchical structure that has been built. Here, a "relevant agent zone", as opposed to an "irrelevant agent zone", means a group whose members contain relevant documents. This is a somewhat fuzzy definition as there is no clear boundary between a relevant-agent-zone and an irrelevant-agent-zone. However, as shown in [9], in reality, most of the agents in the network are irrelevant to a given query. Therefore, in our coordinated search algorithm, we try to locate the "relevant agent zones" by sorting the

$P(Q|G)$ value, which can be considered as the similarity of a query Q and a group collection model G. This value in our paper is again estimated by the similarity between the query and the mediator collection model. Specifically, the query initiator sends out the query to its most similar top-level mediator who then forwards the query to the other top level mediators with a certain TTL(time to live) value. After it expires an agent will not forward the query any further, thus stopping further search along this path. Upon receiving the queries, a mediator will return with the content similarity of the group and the query $P(Q|G)$. The query initiator then picks the N highest similar top-level mediators as the starting points of the second phase search. Notice that a sorting strategy is used to pick the starting points instead of a threshold strategy used in [9] as the calculation for content similarity value between a short query and a collection[6] can be biased by the particular collection properties and thereby making an absolute threshold value inaccurate.

During the second stage, we start the search from the mediators chosen in the first phase. The query initiator forwards the query to the K_{TOP} most promising agents who then proceed to forward the query on to their neighbors in a decreasing order of similarity values. This process continues in the network until all the agents receiving the query drop the message or there are no other agents to forward to. There is no explicit recognition by individual agents that the query is no longer being processed by any agent in the network. The K_{TOP} value affects the number of agents that could be visited when the search algorithm ends. The bigger K_{TOP} value is, the more messages will be generated in the system and potentially more agents will eventually be visited.

5. EXPERIMENT SETUP AND RESULTS ANALYSIS

5.1 Experiment setup

In our experiment, we use TREC-VLC-921 dataset which contains 921 sub-collections to simulate the collections hosted on agents. TREC-VLC-921 was split from TREC VLC1 collection by data sources in order to create testbed for distributed information retrieval research originally[3]. TREC VLC1 is part of the TREC collections which are distributed by the National Institute of Standards and Technology (NIST) for testing and comparing the current text retrieval techniques. TREC VLC1 (very large collection) includes documents from 18 different data sources, such as news, patents, and the Web[4]. We ran the query set 301-350 on TREC-VLC-921 to simulate the user queries. .

We use the algorithm introduced in [7] to calculate the degree limit of the agents with parameters $\alpha = 0.5$ and $\beta = 0.6$. In our experiments, we exp0lored three different upward degree limits and downward degree limits based on linear degree ratios of 0.5, 0.8 and 1.0. During the search process, K_{TOP} value is set to the minimum value of 8 and the number of top-level mediators, this value allows a reasonable coverage of the relevant documents while still keep the traffic low.

In experiments, we build the agent organization incrementally as the 921 agents join the system. Then we ran the search algorithms with the query set $301 - 350$. The query always starts from a randomly picked agent. We define a time unit as the communication time cost from one agent to the next one. Notice that this definition does not include the local search time, which will be analyzed separately in the next section. During the search process, we record the time when each agent is visited and the time when messages are generated. To get a greater than 95 percent confidence interval on the number of messages and CRR value(will be introduced in the next section), we ran the simulation 50 rounds. In each round, we built a new agent organization and repeated every query 50 times on that organization.

The experimental results show that the semantically close agents are consistently grouped together. In this section, we analyze the resulting hierarchical group organization from both an information retrieval and a system performance perspectives. Note that Figures 3, 4, 5 compare the simulation results of the algorithm described in this paper which is denoted as *disc-0.8-ttl-4* in these figures with a random algorithm and the AVRA algorithm. The AVRA algorithm, which is described in our previous work[9], conducts search on a reorganized topology in which agents with similar collection are clustered together implicitly. In Fig.3, The upper line is the centralized approach: Centralized KL divergence based approach is used as an upper bound on performance, which is common in the distributed IR literature. This approach assumes the network is a fully-interconnected graph and agents are visited in the order of decreasing similarity values between collections and each query. In Fig 3, *ImprovedRandom* represents an improved random approach with a heuristic that agents forward queries to the neighboring agents in a decreasing similarity order while the rest of the algorithm is the same as random approach, As it turned out that the degrees of downward and upward links did not contribute much to the system performance in all the three cases, we only keep the curve when the degree ratio is 0.8 in Fig 3, 4 and 5.

5.2 Query processing efficiency

Recall and precision are two important measures in traditional information retrieval research.In this paper, we evaluate the query processing efficiency of our system based on a variant of the recall ratio which is defined below:

Definition 1: Cumulative Recall Ratio (CRR) for a query after n agents are searched is defined as

$$CRR_{q_i,n} = \sum_{j=1}^{n} \frac{r_j}{R_{q_i}}$$

Here R_{q_i} is defined as the total number of relevant documents located in the entire network for the query q_i, and r_j is the number of relevant documents located at agent j.

The traditional recall ratio reflects the proportion of the relevant documents over the retrieved documents in a collection while the CRR value characterizes the relevant documents returned when a certain number of agents have been searched. Therefore, the CRR metric in our system can be considered as the counterpart of recall ratio in traditional information retrieval field. As we assume that each agent is able to distinguish relevant documents from irrelevant ones with 100% precision, the traditional precision ratio corresponds to the capability of the agent selection(database se-

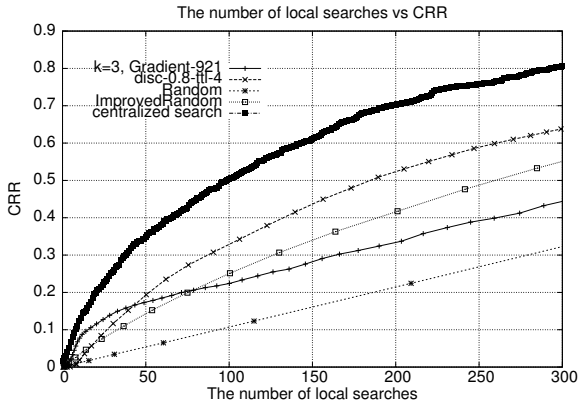

Figure 3: CRR versus the number of local searches

lection) in P2P systems. Thus, in order to maximize the cumulative recall ratio, we need to forward the query to those agents hosting the most relevant documents. Fig. 3 presents experimental data on the ratio of CRR over the agents retrieved. This is achieved by plotting with the pair (CRR, site searched). This measure has theoretical meaning in terms of information retrieval performance as it characterizes the collection selection performance. By searching fewer agents, we are able to significantly reduce message cost and local computational costs.

Note that we are not interested in the exhaustive searches with 100% recall ratio. Instead, we only evaluate the system when a relative small part of the system is searched. This is because that there can be a very large number of relevant information sources in P2P information retrieval as Web information retrieval thereby making an exhaustive search unnecessary. Additionally, exhaustive searches could lead to a bad performance in the presence of many concurrent queries in the system as it consumes more computational resources.

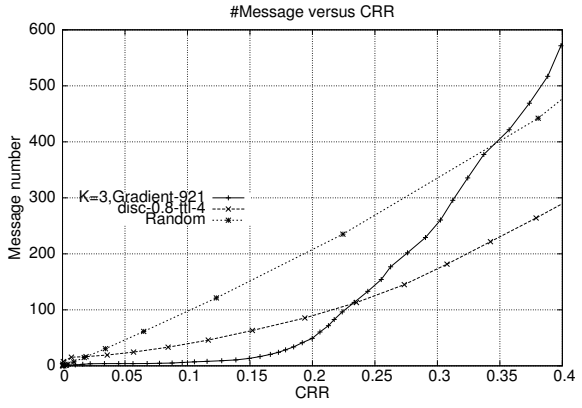

Figure 4: Messages number versus CRR

We have several observations from Fig.3: (1) The central KL approach consistently outperforms the other three approaches. The results are consistent with the conclusion that the collection model is a stable indicator for the collection from distributed information retrieval experiments; (2) As little is known about the content distribution, the number of relevant documents retrieved by random algorithm is proportional to the number of agents that have been searched; (3) The figure demonstrates that the search algorithm on the hierarchical topology consistently outperforms the other approaches including AVRA and the improved random approach; (4) The *ImprovedRandom* algorithm outperforms the AVRA algorithmwhen more agents are visited. As the major difference between the two algorithm is the forwarding strategy when an agent receives a query, this observation indicates that a simple sorting strategy performs better than threshold strategy used in [9]. Again, as we mentioned in the previous sections, we believe the edge of the sorting strategy comes from the fact that absolute content similarity is less accurate in heteregeous environment.

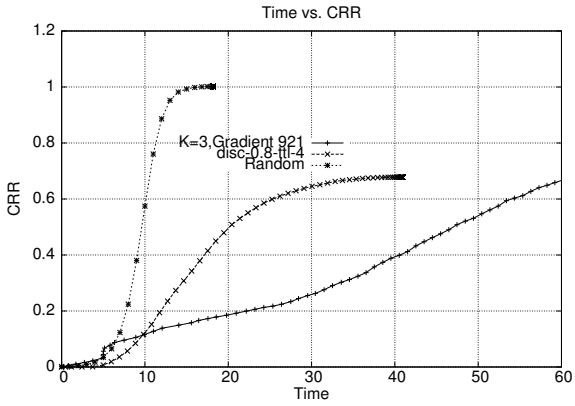

Figure 5: CRR versus communication time

5.3 Traffic and time based efficiency

In addition to the two IR metrics, the number of messages generated and searching time t are two important measures from the system performance perspective. To simplify evaluation, we do not consider the time spent for relevant documents to return to the query initiator. Therefore, we define the the searching time t as the sum of the time spent in the trip from the query initiator to agents, i.e. t_1, and local processing time spent in agents, i.e, t_2. As the local processing involves searching the local document collections, sorting the similarity of neighboring agents to queries and forwarding queries to other agents, we believe that t_2 is the dominant factor in searching time t. Fig. 4 illustrates the number of messages generated versus CRR value. Fig. 5 depicts cumulative recall ratio versus the elapsed time in the search.

Fig.4 illustrates that our approach generates fewer messages than random approach. Particularly when CRR reaches 40%, there are about 500 messages generated in the system for the random search while the two-stage search algorithm generates from $260 - 300$ messages depending on the parameters. An interesting fact is that AVRA algorithm performs very well when CRR is below a certain value, but its performance then drops off sharply above that value. This benefit comes from the fact that AVRA algorithm attempts to locate a good start point for a given query with minimal communication efforts. Therefore, messages number increases slowly at the begining. However, once the query is forwarded out of relevant-agent-zone, AVRA algorithm con-

sumes more messages in order to achieve a certain level of CRR.

Fig.5 demonstrates the time units t_1 spent to reach a certain CRR value. Not surprisingly, with the presence of only a single query, the two-stage search algorithm and AVRA algorithm spend more time in communication. This fact is attributed to the facts that (1) the two-stage search is a focused search approach while the random search algorithm broadcasts the query to every possible directions.; (2) In AVRA algorithm, as agents of similar collection models are normally clustered together, a query tends to be forwarded to same agent, thereby making the query harder to reach more agents. However, we believe that in real system, both AVRA and the two-stage search algorithm would perform much better. This confidence comes from two reasons: (1) as we mentioned in the previous section, we believe that the t_1 is a minor factor contributing to the search time t considering the agent selection efficiency. (2) Fig.5 only shows the situation with a single query in the system. With many queries concurrently in the system, considering the messages queuing time, the situation in Fig 5 may not be an accurate predictor of performance. We leave this simulation as future work.

6. RELATED WORK

Peer-to-peer (P2P) systems have emerged as a popular way to share huge volumes of data. However, early work in this area focused on file-sharing systems with exact-match based searching approaches. Most recently, P2P based information retrieval, which allows partial match among documents and queries, has attracted significant attention.Among these works, many researchers are also working on the combination of underlying topologies and searching schemes in order to improve system performance[5] [2]. In [5], the authors use a vector model to represent each node and propose a clustering algorithm to cluster agents with similar content together through the "attractive" links. Based on this model, the authors further propose a *Firework* query model. This work is very similar to our previous work in which content-similar agents are implicitly clustered together by aggregating collection models for agents and two search strategies are employed based on the reorganized topology[9]; In [2], the authors show that a simple search strategy on possession-rule overlays can dramatically increase the effectiveness of search for rare items over that of plain unstructured network, and the performance can be further improved by preferring rules that correspond to recently acquired items or rules where the meta data of the corresponding items is more related to the query items.

Our work differs from these works in that our topology algorithm forms a multi-level hierarchical structure which can better reflect the inherent connections among those collections. We also believe that our model has better scalability in larger content sharing networks. In our algorithm, the agents calculate content-similarity thresholds offline and thus we do not need a pre-set topic number and level for the topic hierarchy. Additionally, we proposed a context-aware searching algorithm to take advantage of the hierarchy structure to reduce the traffic and improve throughput.

7. CONCLUSION

In the multi-agent based content sharing system, the flat, peer-to-peer(P2P) organization hinders agents from efficiently locating relevant agents for queries. This paper develops and analyzes a hierarchical agent group formation protocol to build a hybrid organization for a large-scale content sharing system in an incremental and distributed fashion. During the group formation process, agents are classified into various semantic groups by organizing their agent view structure.A collection model based coordinated search algorithm is also proposed to take advantage of the organization. We evaluated the system performance based on TREC-VLC-921 datasets. The results of the experiments demonstrate a significant increase in the cumulative recall ratio(CRR) measure compared to the flat agent organization and structure.

8. REFERENCES

[1] J. Callan. *Distributed information retrieval.* Kluwer Academic Publishers, Reading, Massachusetts, 2000.

[2] E. Cohen, A. Fiat, and H. Kaplan. Associative search in peer to peer networks: Harnessing latent semantics. In *In Proceedings of the IEEE INFOCOM'03 Conference*, 2003.

[3] J. French, A. Powell, J. Callan, C. Viles, T. Emmitt, K. Prey, and Y. Mou. Comparing the performance of database selection algorithm. In *Proceedings of the 22nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1999.

[4] D. Hawking, N. Craswell, and P. Thistlewaite. Overview of trec-6 very large collection track. In *In Proceedings of the Tenth Text Retrieval Conference TREC*, pages 93–105, 1997.

[5] C.-H. Ng and K.-C. Sia. Peer clutering and firework query model. In *Proceedings of the 11th World Wide Web Conference, Poster*, 2002.

[6] P. Ogilvie and J. Callan. Experiments using the lemur toolkit. In *In the Tenth Text Retrieval Conference, TREC 2001. NIST Special Publication*, pages 103–108, 2001.

[7] C. R. Palmer and J. G. Steffan. Generating network topologies that obey power laws. In *Proceedings of GLOBECOM '2000*, November 2000.

[8] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281. ACM Press, 1998.

[9] H. Zhang, W. Croft, B. Levine, and V. Lesser. A multi-agent approach for peer-to-peer information retrieval. In *Proceedings of Third International Joint Conference on Autonomous Agents and Multi-Agent Systems*, July 2004.

[10] H. Zhang and V. Lesser. A dynamically formed hierarchical agent organization for a distributed content sharing system. In *University of Massachusetts, Amherst Technical Report UM-CS-04-47.*