Design and Implementation of Agent Community based Peer-to-Peer Information Retrieval Method

Tsunenori Mine[†], Daisuke Matsuno[‡], Akihiro Kogo[‡], and Makoto Amamiya[†]

{Faculty[†], Graduate School[‡]} of Information Science and Electrical Engineering,
Department of Intelligent Systems, Kyushu University
6-1 Kasuga-koen, Kasuga, Fukuoka 816-8580, Japan
{mine,kogo,amamiya}@al.is.kyushu-u.ac.jp,
WWW home page: http://www-al.is.kyushu-u.ac.jp/~mine

Abstract. This paper presents an agent community based peer-to-peer information retrieval method called ACP2P method[16] and discusses the experimental results of the method. The ACP2P method uses agent communities to manage and look up information related to users. An agent works as a delegate of its user and searches for information that the user wants by communicating with other agents. The communication between agents is carried out in a peer-to-peer computing architecture. In order to retrieve information relevant to a user query, an agent uses a content file, which consists of retrieved documents and two histories : a query/retrieved document history(Q/RDH) and a query/sender agent history(Q/SAH). The former is a list of pairs of a query and the address of an agent that returned documents relevant to the query. The latter is a list of pairs of a query and the address of a sender agent and shows "who sent what query to the agent". This is useful for finding a new information source. Making use of Q/SAH is expected to have a collaborative filtering effect, which gradually creates virtual agent communities, where agents with the same interests stay together. Our hypothesis is that a virtual agent community reduces communication loads necessary to perform a search. As an agent receives more queries, then more links to new knowledge are acquired. From this behavior, a "give and take"(or positive feedback) effect for agents seems to emerge. We implemented this method with Multi-Agent Kodama, and conducted experiments to test the hypothesis. The experimental results showed that the method employing two histories was much more efficient than a naive method employing 'multicast' techniques only to look up a target agent. Further, making use of Q/SAH facilitates bidirectional communications between agents and thus creates virtual agent communities.

1 Introduction

The rapid growth of the World Wide Web has made conventional search engines suffer from decreasing coverage in searching the Web. Internet users meet information floods every day, and are forced to filter out and choose the information

they need. In order to deal with these problems, a lot of studies on distributed information retrieval(e.g. [4, 3]), information filtering(e.g. [14]), information recommendation (e.g. [22]), expert finding(e.g. [28],[12]), or collaborative filtering (e.g. [9],[18],[10],[21]) have been carried out. Most systems developed in that research are, unfortunately, based on the server-client computational model and are often distressed by the fundamental bottle-neck coming from their central control system architecture. Although some systems based on the peer-to-peer (P2P for short) computing architecture (e.g. [24],[5],[8],[17]) have been developed and implemented, each node of most those systems only deals with simple and monolithic processing chores.

Considering these issues, we proposed an Agent Community based P2P information retrieval method (ACP2P method for short)[16]. The ACP2P method uses agent communities to manage and look up information related to a user query. An agent works as a delegate of its user and searches for information that the user wants by communicating with other agents. The communication between agents is carried out based on a P2P computing architecture. In order to retrieve information relevant to a user query, an agent uses two histories : a query/retrieved document history(Q/RDH for short) and a query/sender agent history(Q/SAH for short). The former is a list of pairs of a query and the address of the agent that returned documents relevant to the query, where the query was sent by the agent itself. The latter is a list of pairs of a query and a sender agent and shows "who sent what query to the agent". This is useful for finding a new information source. Making use of the Q/SAH is expected to make a collaborative filtering effect emerge and to gradually create virtual agent communities, where agents with the same interests stay together. Our hypothesis is that a virtual agent community reduces communication loads necessary to perform a search. As an agent receives more queries, then more links to new knowledge are acquired. From this behavior, a "give and take"(or positive feedback) effect for agents seems to emerge. We implemented the method with Multi-Agent Kodama and conducted experiments to test the hypothesis, i.e., to evaluate how well Q/SAH works for reducing communication loads and for making a "give and take" effect emerge. This paper presents the ACP2P method and discusses the details of experimental results. The results showed that the method reduced communication loads much more than other methods which do not employ Q/SAH to look up a target agent, and was useful for making a "give and take" effect.

The remainder of the paper is structured as follows. Section 2 considers the ACP2P method. Section 3 discusses the experimental results and Section 4 describes related work.

2 Agent Community based Peer-to-Peer Information Retrieval Method

2.1 Overview of ACP2P Method

The ACP2P method employs three types of agents: user interface(UI) agent, information retrieval(IR) agent and history management(HM) agent. A set of

three agents (UI agent, IR agent, HM agent) is assigned to each user. Although a UI agent and an HM agent communicate only with the IR agent of their user, an IR agent communicates with other users' IR agents not only in the community it belongs to, but also in other communities, to search for information relevant to its user's query. A pair of Q/RDH and Q/SAH histories is managed by the HM agent.

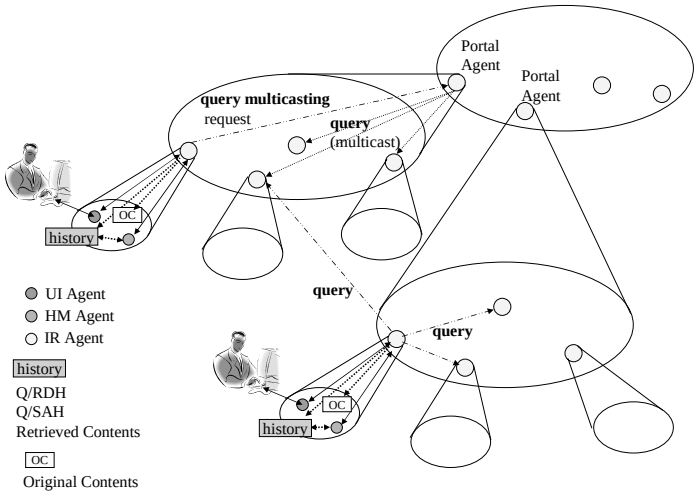

Fig. 1. Agents and their Community Structure

Fig. 1 shows an example of the agent community structure which the ACP2P method is based on. A portal agent in the figure is the agent which is a representative of a community and manages all member agents' addresses there, where a member agent of a community designates an IR agent. Unlike a super-peer of a super-peer network[27], the portal agent originally does not behave as a server of all member agents in the community, but mediates between a member agent and the others for advertising its joining the community or telling its messages to them.

When a member agent wants to find any target agents which have information relevant to a query, the agent looks them up using a content file, which consists of retrieved documents, and two histories: Q/RDH and Q/SAH. The format of the content file and two histories will be described in the next section.

If the target agents are found, a query is sent directly to them, and their retrieved results are also returned directly to the query sender IR agent. If the requested number of such agents is not found, the agent asks the portal agent to send the query to the all member agents in the community by a multicast technique. At that time, all the answers will be returned to the portal agent. If the number of results with a 'YES' message reaches the requested number, without waiting for the rest of answers by other IR agents, the portal agent sends

them back to the query sender IR agent. Even if the number of 'YES' messages did not reach the requested number after all IR agents replied, the portal agent also sends the currently held results to the query sender IR agent.

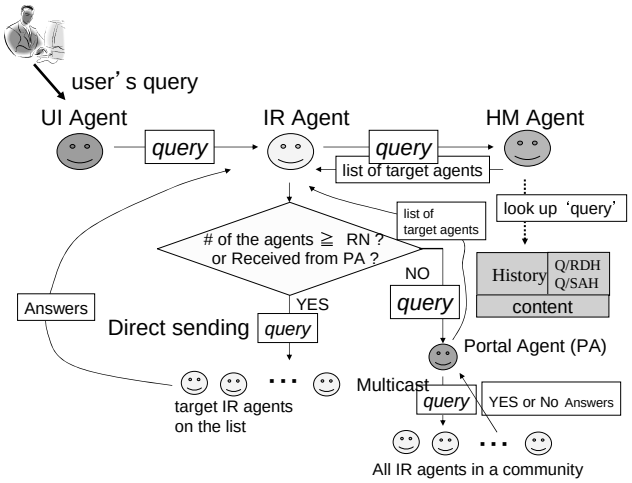

Fig. 2. Actions for Sending a Query

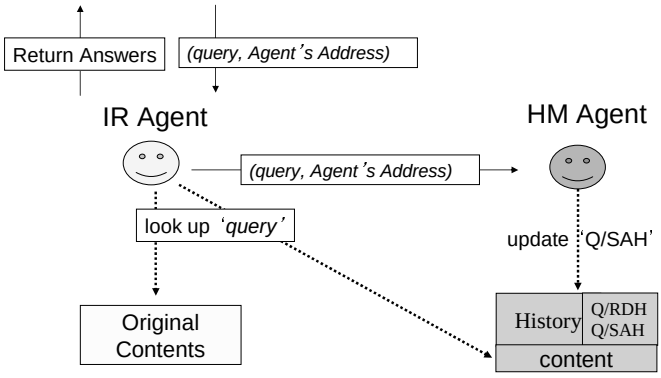

Fig. 3. Actions for Receiving a Query

Fig. 2, 3 and 4 show the processes or data flows in the following three cases : 1) an IR agent sends a query, 2) an IR agent receives a query from another IR agent or a portal agent, and 3) an IR agent receives answers from other IR agents, respectively. When receiving a query from a UI agent, an IR agent asks an HM agent to look up target agents with its history or a portal agent to do it using a query multicasting technique (Fig. 2).

When receiving a query from other IR agents, the IR agent looks up the information relevant to a query, sends an answer to the query sender IR agent, and sends a pair of a query and the query sender IR agent's address to an HM agent so that it can update Q/SAH (Fig. 3).

The returned answer to the query is either a pair of a 'YES' message and retrieved relevant documents or a 'No' message, which represents no relevant information, provided that retrieved documents are not returned when the query comes through a portal agent.

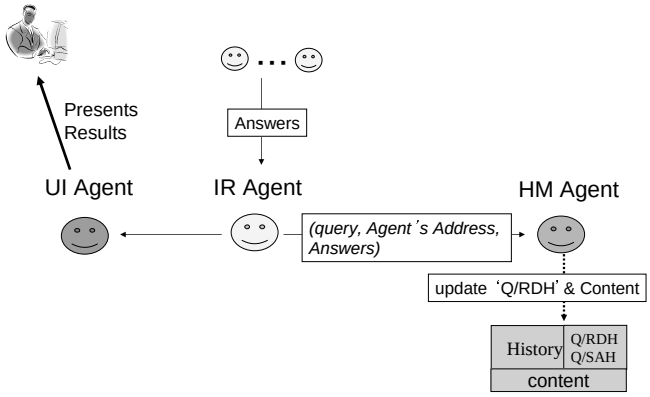

Fig. 4. Actions for Receiving Answers

When receiving answers with a 'YES' message from other IR agents, an IR agent sends them to a UI agent, and sends them with a pair of a query and the addresses of answer sender IR agents to an HM agent to update Q/RDH (Fig. 4).

2.2 Content file and History files

Table 1 shows the formats of a document content file: **Content** and two histories: **Q/RDH** and **Q/SAH**. The document content file consists of a list of 4-tuples <**title, text, original, range**>, namely, the title of a document, its text content, the address of the IR agent whose user owns the document, and the allowed distribution range of the document, respectively. Documents retrieved and returned by other IR agents are shared into the Content file without any redundant registration. Thus the same content returned by two or more IR agents is registered only once into the Content file. Original documents, which are created by a user and initially assigned to his/her agent, also take the same format as the Content.

The Q/RDH file comprises a list of pairs of <**query, from**>, each of which is a query sent by the agent itself and the address of IR agent that returned this retrieved information, respectively. The Q/SAH file is a list of pairs <**query,**

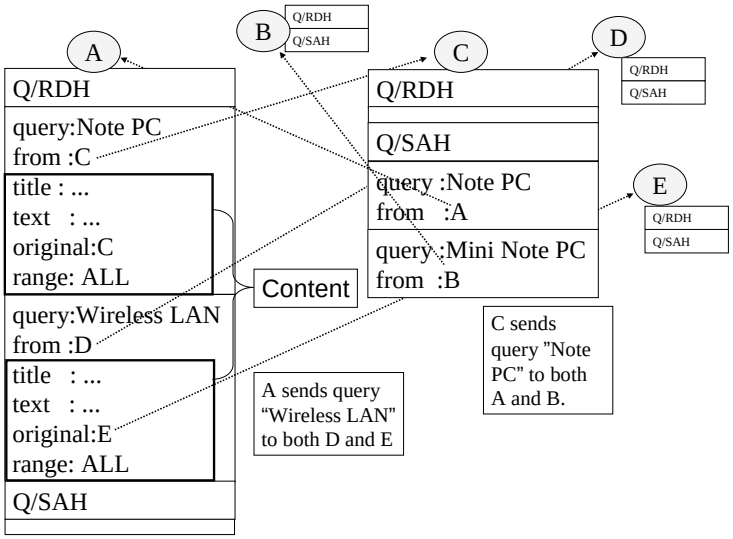

Fig. 5. Example to find target IR agents using two histories and content file. A to E in circles represent IR agents' names.

from>, each of which is a query and the address of the agent which sent the query to the IR agent. Table 2 shows an example of part of a document content file. Table 3 also shows an example of part of Q/SAH file, which was originally written in Japanese.

2.3 Determining Target Agents using both Histories

In order to determine the target agents to send a user query, an IR agent uses the contents of retrieved document files and two histories, Q/RDH and Q/SAH. Fig. 5 depicts an example how the target agents are found, where Ⓐ to Ⓔ represent IR agents. For simplicity, we assume here that the IR agent does the job of an

Table 1. The structures of Content file and two histories: Q/RDH and Q/SAH

Content	title	the title of document
	text	the content of document
	original	the address of the IR agent whose user created the document
	range	the range allowed to be distributed(ALL, Community, Agent)
Q/RDH	query	queries sent by the agent recorded in the from field
	from	the address of IR agent which has replied to the query in the query field
Q/SAH	query	queries sent by the agent recorded in the from field
	from	the address of the IR agent who sent the query in the query field

Table 2. A part of content file

title	text	original	range
Netscape informal FAQ Japanese version	HTML text in the file	com_Netscape@	ALL

Table 3. A part of Q/SAH

query	from
telegram	root.p2p.com_telegram@
treatment	root.p2p.sic_hepatitis_type_C@
Asthma	root.p2p.sic_Asthma@
Human	root.p2p.sic_Adult_Children@
Thing	root.p2p.sic_Alzheimer's_Disease@
Ill	root.p2p.sic_Jacob_Disease@
Dream	root.p2p.sic_Dealer@
Mastocarcinoma	root.p2p.sic_Mastocarcinoma@
Hoof	root.p2p.sic_Hoof-and-Mouth-Disease@

HM agent. Furthermore, to show the correspondence between a query and a retrieved document, we show the content files in Q/RDH.

Ⓐ has two query entries in its Q/RDH. Both queries were sent by Ⓐ itself. This figure shows that Ⓐ sent query 'Note PC' to Ⓒ and got the retrieved results from Ⓒ. Ⓒ recorded the query and Ⓐ's address into its Q/SAH. Since Ⓐ received the results from Ⓒ, Ⓒ's address was recorded in the 'from' field of the same record as the query in Ⓐ's Q/RDH. In addition, since the content included in the results is the original of Ⓒ's user, Ⓒ's address is seen in the 'original' field of the content. In the same way, Ⓐ also sent query 'Wireless LAN' to Ⓓ, Ⓓ returned retrieved documents to it, and Ⓓ's address was recorded into the 'from' field of the same record as query 'Wireless LAN' in Ⓐ's Q/RDH. Since the documents include a content created by Ⓔ's user, Ⓔ's address is seen in the 'original' field of the content.

After getting these histories, if Ⓐ sends another query which is similar to 'Wireless LAN', say 'LAN', Ⓐ not only can find Ⓓ in a 'from' field of Q/RDH, but also find Ⓔ from an 'original' field of the content file by calculating a similarity between the query and the content file. Accordingly Ⓐ sends the query to both Ⓓ and Ⓔ.

The figure also shows that Ⓒ received query 'Mini Note PC' from Ⓑ, and both the query and Ⓑ's address were recorded into the Q/SAH. Even if Ⓒ has not sent a query, it can find information related to the queries it received using its Q/SAH. Therefore when Ⓒ sends a query, say 'Note PC', it will find Ⓐ and Ⓑ with the Q/SAH and can consequently send the query to them.

2.4 The Effect of both Histories

As mentioned in the previous section, both Q/RDH and Q/SAH help to find target agents to send a query to. If an IR agent can find a sufficient number of

agents, no 'query multicasting' is carried out. Both histories, consequently, help to reduce communication loads between agents.

The user's positive or negative judgments concerning the retrieved results could be embedded into them in Q/RDH. These user evaluations are expected to be useful for finding target agents which will return relevant information, creating a collaborative filtering effect. As a user creates more information, his/her IR agent can return the retrieved results to more queries. Such an IR agent consequently receives more queries from other agents. Thus, the agent accumulates more information sources comprised of pairs of a query and a sender agent's address in its Q/SAH. That leads to the emergence of a 'give and take' effect.

2.5 Overview of Multi-Agent Kodama

The ACP2P method was implemented with Multi-Agent Kodama (Kyushu university Open & Distributed Autonomous Multi-Agent) [29]. Kodama comprises hierarchical structured agent communities based on a portal-agent model. A portal agent(PA) is the representative of all member agents in a community and allows the community to be treated as one normal agent outside the community. A PA has its role limited in a community, and the PA itself may be managed by another high-level portal agent. A PA manages all member agents in a community and can multicast a message to them. Any member agent in a community can ask the PA to multicast its message. All agents form a logical world which is completely separated from the physical world consisting of agent host machines. That means agents are not network-aware, but are organized and located by their places in the logical world. This model is realized with the agent middle-ware called Agent Communication Zone(ACZ for short). ACZ is primarily designed to act as a bridge between distributed physical networks, creating an agent-friendly communication infrastructure on which agents can be organized in a hierarchical fashion more easily and freely. ACZ is also designed to realize a peer-to-peer communication between agents.

A Kodama agent consists of a kernel unit and an application unit. The kernel unit comprises the common basic modules shared by all Kodama agents, such as the community contactor or message interpreter. The application unit comprises a set of plug-in modules, each of which is used for describing and realizing a specialized or original function of agents. For more details, please see [29].

3 Experiments

3.1 Preliminaries

We used the Web pages of Yahoo! JAPAN[26] for the experiments. The Web pages used are broadly divided into five categories: animals, sports, computers, medicine, and finance. Each of them consists of 20 smaller categories, which are selected in descending order of the number of Web pages recorded in a category. An IR agent is assigned to each selected category, and thus 100 IR agents are

created and activated in the experiments. A category name is used as the name of an IR agent, and the Web pages in the category are used as the original documents of the agent. All agents are realized by implementing their functions in plug-in modules of Kodama's application unit.

Each IR agent sends 10 queries, which all belong to either query set QL=1 or QL=2. QL=1 and QL=2 consist of 10 queries, whose query length is one and two, respectively, where query length means the number of terms in a query. When using queries belonging to QL=1, 10 nouns are extracted from every category assigned to each IR agent in descending order of their frequency of occurrence in the category. Each of the nouns is used as a query of the IR agent. When using those belonging to QL=2, 5 nouns are extracted, and the combinations of the extracted 5 nouns taken in pairs create 10 queries.

All IR agents were assigned to the same community for simplicity.

We conducted experiments to show how two histories help to reduce communication loads between agents looking for information relevant to a query, and how Q/SAH helps in searching for new information sources. To perform the experiments, we compared three methods : 1) ACP2P with a Q/SAH(wQ/SAH for short), 2)ACP2P without a Q/SAH(woQ/SAH for short), and 3) Simple method always employing a 'multicast' technique (MulCST for short).

3.2 Similarity Measure for Information Relevant to a Query

In order to find the requested number of target agents to be sent a query, we calculate $Score(query, t\_agent)$, which returns the similarity value between query $query$ and target agent t_agent, with equation (1); $Score(query, t\_agent)$ becomes higher if t_agent sends a greater number of similar queries and returns more documents related to $query$. After calculating $Score(query, t\_agent)$ for each IR agent t_agent in the Content file and both histories : Q/RDH and Q/SAH, the requested number (RN) of target agents will be selected in the descending order of $Score(query, t\_agent)$, which value should be more than 0. Whenever the RN of agents is not found, the 'query multicasting' technique will be employed by a portal agent. At that time, all answers will be returned to the portal agent.

If a target IR agent finds information relevant to $query$ from its Content files, it returns a 'YES' message, otherwise a 'NO' message as mentioned in section 2.1. The judgment as to whether or not a document is relevant to a query is made according to the criterion of Boolean AND matching, that is, if the document includes the conjunctions of all terms in $query$, it will be judged relevant, otherwise irrelevant.

$$Score(query, t\_agent) = \sum_{i=1}^{k} \cos(\boldsymbol{query}, \boldsymbol{qh_{di}}) + \sum_{i=1}^{m} (\cos(\boldsymbol{query}, \boldsymbol{qh_{sa_i}}) + \varphi(i)) + \sum_{i=1}^{n} Sim_d(\boldsymbol{query}, \boldsymbol{doc_i}) \quad (1)$$

$$\varphi(i) = \begin{cases} \delta & \text{if } qh_{sa\,i} \text{ is the query sent by other IR} \\ & \text{agent directly.} \\ 0 & \text{otherwise} \end{cases}$$

In equation (1), *query* consists of $w_1, ..., w_m$, where w_i $(1 \leq i \leq m)$ is a term in *query*. qh_d and qh_{sa} represent a query in a record of Q/RDH and Q/SAH, respectively. The first term $\sum_{i=1}^{k} \cos(\boldsymbol{query}, \boldsymbol{qh_{di}})$ returns the total score of the similarities between *query* and each of k number of queries sent to *t_agent*. The second term $\sum_{i=1}^{m}(\cos(\boldsymbol{query}, \boldsymbol{qh_{sai}}) + \varphi(i))$ represents the score between *query* and $qh_{sa\,i}$, which is the *i_th* of m queries sent by *t_agent* in Q/SAH. $\varphi(i)$ is a weight to consider the importance of 'direct sending of a query.' If $qh_{sa\,i}$ is sent directly by *t_agent*, δ is added to the score. The last term $\sum_{i=1}^{n} Sim_d(\boldsymbol{query}, \boldsymbol{doc_i})$ is the total score of similarities between *query* and each of n documents originally created or just owned by the user of *t_agent*. $Sim_d(query, doc)$ represents the similarity between *query* and the content of retrieved document *doc*. It is calculated with the following equation, which is a simplified form of BM15[20].

$$Sim_d(query, doc) = \sum_{i=1}^{m} \frac{tf_i}{tf_i + 1}$$

Where tf_i represents the frequency of occurrence of w_i in *doc*.

The reason why we did not consider a "inverse document frequency" factor on the equation is based on preliminary experimental results.

3.3 Experimental Results

First, in order to decide the value of δ in $qh_{sa\,i}$ of equation 1, and to broadly figure out whether or not Q/SAH helps in looking for new information sources and in creating virtual communities, we compared three methods : woQ/SAH, wQ/SAH($\delta = 0$) and wQ/SAH($\delta = 0.1$). wQ/SAH($\delta = 0$) represents the method wQ/SAH with δ=0 and wQ/SAH($\delta = 0.1$) is the method wQ/SAH with δ=0.1. The results are shown in table 4, and show that both methods adopting Q/SAH are better than woQ/SAH. In addition, wQ/SAH($\delta = 0.1$) was slightly better than wQ/SAH($\delta = 0$) from the points of view of reducing the number of messages exchanged, increasing the number of achieved results and the number of agents making bidirectional communications. From these results, we decided to adopt wQ/SAH($\delta = 0.1$) as wQ/SAH.

Next, to show how much wQ/SAH works for reducing communication loads, we investigated the change of the average number of messages exchanged by each IR agent for every query input. The experiments were conducted with two query sets: QL=1 and QL=2, on which tests with 4 different requested numbers : RN=3, 5, 7 and 10 were performed. The results are shown in Fig. 6. In both cases, the average number of messages exchanged by each IR agent is reduced for every additional query input, and increases as RN does.

Table 4. The effect of δ for wQ/SAH

RN=5	woQ/SAH	wQ/SAH($\delta = 0$)	wQ/SAH($\delta = 0.1$)
# of Messages	46112	37361	30050
# of retrieved results	9223	13673	14014
# of One way agents	307	245	206
# of pairs of Bidirectional agents	144	272	326
The retrieval failure ratio	10.79%	8.44%	11.36%

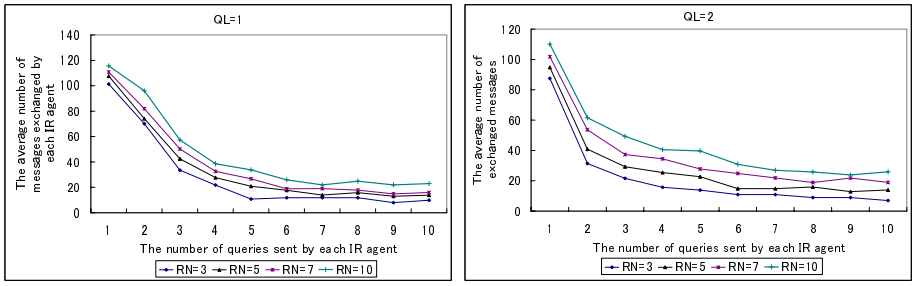

Fig. 6. The average number of messages exchanged by each IR agent for each query input, every query belongs to QL=1(left) or to QL=2(right)

Further, for both QL=1 and QL=2, we compared the three methods: wQ/SAH, woQ/SAH and MulCST. RN was set to 10. The results are shown in Fig. 7. In both cases, the number of exchanged messages in MulCST did not change for every query input, while that for both wQ/SAH and woQ/SAH was reduced. In addition, wQ/SAH had better performance than woQ/SAH.

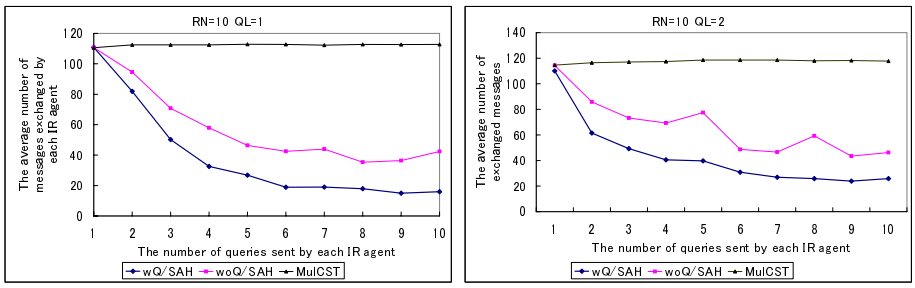

Fig. 7. The average number of messages exchanged by each IR agent for each query input, where QL=1 is the left, and QL=2 the right. RN=10 in both cases.

We also compared three methods on the average number of documents acquired by each IR agent. The results are shown in table 5. Except for the case of RN=3 of QL=2, there was little difference between wQ/SAH and MulCST.

Table 5. Comparison on average number of acquired documents when query length is 1 (left) and 2 (right)

QL=1, RN=	3	5	7	10
wQ/SAH	269.1	385.3	443.0	497.6
woQ/SAH	258.8	331.6	424.9	476.4
MulCST	233.8	366.4	421.3	487.0

QL=2, RN=	3	5	7	10
wQ/SAH	54.9	126.3	178.9	226.8
woQ/SAH	54.8	96.8	150.0	208.3
MulCST	85.3	148.4	191.0	232.6

Fig. 8 compares the three methods on the content acquisition efficiency, that is, the average number of acquired documents per one exchanged message in the cases of RN=3, 5, 7 and 10, for QL=1 and QL=2. For both query sets, wQ/SAH had the best performance, which was for RN=7 in QL=1 and RN=10 in QL=2.

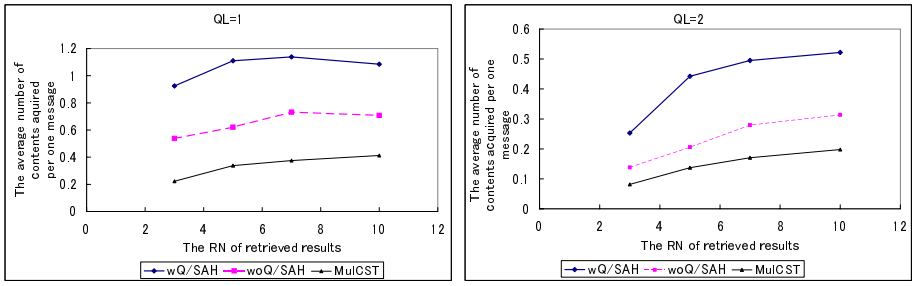

Fig. 8. Comparison of Content Acquisition Efficiency

Further, we investigated the failure ratio caused by wQ/SAH and woQ/SAH because making use of query histories was just a good heuristic and might have failed to select appropriate target agents. The experiment was carried out with QL=2 because in the case of QL=1, no failure will occur since every query sent by each IR agent is different and the query is created from the original documents of the agent. The experimental results are shown in Fig. 9.

As the value of RN increases, the failure ratio decreases. This is because, as the value of RN increases, the number of acquired documents also increases as shown in table 5, and thus the number of information sources also increases. Consequently, it becomes easier to select target agents having information relevant to a query. As a baseline, we investigated the ratio of the number of agents returning a 'YES' message to each query, to the total number of IR agents. It was about 20%. That means when target agents are randomly selected, about 80% retrieval failure can occur. Considering that the retrieval failure ratio for both wQ/SAH and woQ/SAH was less than 20%, we can say that making use of two histories is effective in finding information relevant to user queries.

Lastly, we conducted an experiment to show how the number of agents exchanging query messages together increases, comparing wQ/SAH and woQ/SAH. The results are shown in Fig. 10. In the case of wQ/SAH, the number of pairs of

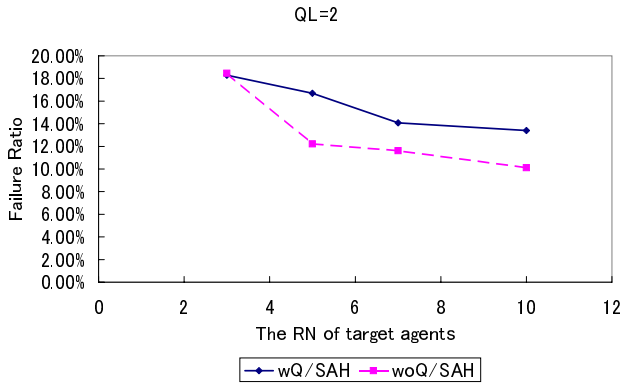

Fig. 9. Failure Ratio for each RNof target agents, QL=2

agents making bidirectional communication is much greater than that of those making one way communication. On the other hand, woQ/SAH shows the opposite tendency. These results show that Q/SAH facilitates bidirectional communications and then creates virtual agent communities where agents exchanging similar queries, i.e., having the same interests, stay together.

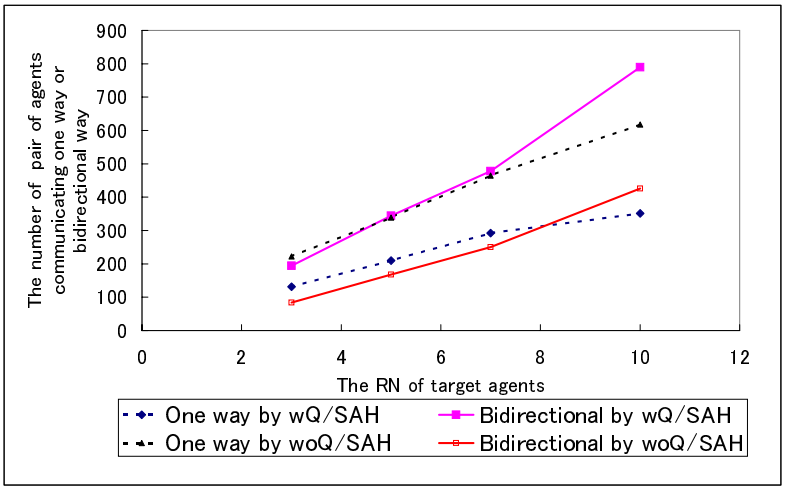

Fig. 10. The change of number of pairs of agents making bi-directional or One-way communication, for QL=2

4 Related Work

There is lots of work related to the topics touched on in this paper, such as distributed information retrieval(DIR), P2P file searching, collaborative filtering and so forth. DIR selects some IR systems to send a query, aggregates the results

returned by the selected IR systems, and presents them to a user. Before selecting the IR systems to be sent a query, the resource description of each IR system is often created[3]. In the ACP2P method, Q/RDH incrementally creates an effect similar to resource description, and furthermore, Q/SAH works as a good heuristic like collaborative filtering in finding relevant information.

A lot of P2P file searching systems such as Freenet[5], Chord[24], Gnutella[8], Napster[17] and pSearch[25] have been proposed. Freenet and Chord are carried out in a pure P2P computing architecture. They neither employ 'broadcast' techniques like Gnutella, nor have a centralized server machine like Napster. Freenet provides information sharing and information finding functions among anonymously distributed nodes. Although Chord does not provide anonymity of nodes, it has an efficient protocol for looking up nodes. Their node searching strategies are conducted according to keywords attached to the information of the nodes. On the other hand, the ACP2P method makes use of the content information of documents, and two histories: Q/RDH and Q/SAH to search for target agents with relevant information. In particular, Q/SAH provides similar effects to link analysis like PageRank[2] or HITs algorithm[13] and makes a natural collaborative filtering effect emerge. pSearch[25] realizes a semantic overlay network on physical nodes in a content-addressable network(CAN)[19] by distributing document indices based on document semantics, which are generated by Latent Semantic Indexing(LSI)[6]. The search cost for a given query is thereby reduced, since the indices of semantically related documents are likely to be co-located in the network[25]. However, since LSI requires a document-term matrix, pSearch initially needs to collect all documents from every node.

I-Gaia[7] is an application layer for information processing in the DIET architecture, which is a Multi-Agent System development platform. It is formed of three types of agents: s-infocytes(SI), m-infocytes(MI) and t-infocytes(TI). It defined informaion-pull and push tasks with Reuters text-classification corpus and showed their results. For both tasks, queries sent by SI and documents published by MI reach MIs and SIs through TI, respectively. Thus, adequate routing between SIs and MIs is learned by TI. On the other hand, the ACP2P method does not rely on a special agent like TI for sending queries or publishing documents, but each IR agent directly communicates with other IR agents based on a peer-to-peer communicating method.

Although lots of work on the field of Collaborative Filtering (e.g.[9],[18],[23],[1] ,[11],[15],[21]) has been done, most of it assumes the server-client computational model and needs a procedure to collect all data from other nodes explicitly. The ACP2P method takes a distributed data management method with agent communities based on a P2P computing architecture, and makes a natural collaborative filtering effect emerge, with both Q/RDH and Q/SAH.

5 Conclusion and Future Work

We discussed an agent community based information retrieval method, called the ACP2P method, which used the content of retrieved document files and two

histories: Q/RDH and Q/SAH to find target agents to be sent a query. The method was implemented with Multi-Agent System Kodama.

We conducted several experiments to show whether or not two histories helped to reduce communication loads between agents in searching for information relevant to a query, and whether or not Q/SAH helped in looking up new information sources. The experimental results showed the efficiency of ACP2P method and the usefulness of two histories for looking up new information sources. We also investigated and confirmed that the number of agents exchanging query messages together was increased by Q/SAH.

We are currently investigating how to measure the accuracy of or to rank retrieved results, and considering how we can make use of user feedback embedded into the results. Developing an effective method for creating hierarchical agent communities to allocate agents to at the initial stage is future work.

Acknowledgment

We thank Dr. Ken'ichi Takahasi, Satoshi Amamiya and Dr. Guoqiang Zhong for their comments of this early work and support of Kodama System.

This research was supported by the Japan Society for the Promotion of Science under the Grant-in-Aid for Scientific Research (A) No. 15200002 and (C) No. 16500082.

References

1. M. Balabanovic and Y. Shoham. Content-based, collaborative recommendation. *Communications of the ACM*, 40(3), 1997.
2. S. Brin and L. Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. In *Proc. of 7th International World Wide Web Conference:WWW7 Conference*, 1998.
3. J. Callan and M. Connell. Query-based sampling of text databases. *ACM Transactions on Information Systems*, 19(2):97–130, 2001.
4. J. Callan, M. Connell, and A. Du. Automatic discovery of language models for text databases. *ACM SIGMOD*, pages 479–490, 1999.
5. I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. *Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability, http://www.doc.ic.ac.uk/~twh1/academic/*, 2001.
6. S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 1990.
7. A. Gallardo-Antolin et al. I-Gaia : an information processing layer for the diet platform. In *the first international joint conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, pages 1272–1279, 7 2002.
8. Gnutella. http://gnutella.wego.com/, 2000.
9. D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35:61–70, 1992.

10. N. Good, J. B. Schafer, J. A. Konstan, A. Borchers, B. M. Sarwar, J. L. Herlocker, and J. Riedl. Combining collaborative filtering with personal agents for better recommendations. In *AAAI/IAAI*, pages 439–446, 1999.
11. J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR99*, pages 230–237, 1999.
12. A. Kanfer, J. Sweet, and A. Schlosser. Humanizing the net: Social navigation with a "know-who" email agent. In *The 3rd Conference on Human Factors & The Web, http://www.ncsa.uiuc.edu/edu/trg*, 1997.
13. J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46:604–632, 1999.
14. K. Lang. NewsWeeder: learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning*, pages 331–339. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1995.
15. P. Melville, R. J. Mooney, and R. Nagarajan. Content-boosted collaborative filtering. In *SIGIR-2001 Workshop on Recommender Systems*, 2001.
16. T. Mine, D. Matsuno, K. Takaki, and M. Amamiya. Agent community based peer-to-peer information retrieval. In *the third international joint conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, 7 2004. poster.
17. Napster. http://www.napster.com/, 2000.
18. P.Resnick, N.Iacovou, M.Suchak, P.Bergstrom, and J.Riedl. Grouplens: Open architecture for collaborative filtering of netnews, 1994.
19. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *SIGCOMM*, pages 161–172, 2001.
20. S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1994.
21. B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW10*, pages 285–295, 2001.
22. J. B. Schafer, J. A. Konstan, and J. Riedi. Recommender systems in e-commerce. In *ACM Conference on Electronic Commerce*, pages 158–166, 1999.
23. U. Shardanand and P. Maes. Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, volume 1, pages 210–217, 1995.
24. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications*, pages 149–160, 2001.
25. C. Tang, Z. Xu, and S. Dwarkadas. Peer-to-peer information retrieval using self-organizing semantic overlay networks. In *SIGCOMM*, 2003.
26. Yahoo. http://www.yahoo.co.jp/, 2003.
27. B. Yang and H. Garcia-Molina. Designing a super-peer network. In *IEEE International Conference on Data Engineering*, 3 2003.
28. D. Yimam-Seid and A. Kobsa. Expert finding systems for organizations: Problem and domain analysis and the demoir approach. *Journal of Organizational Computing and Electronic Commerce*, 13(1):1–24, 2003.
29. G. Zhong, S. Amamiya, K. Takahashi, T. Mine, and M. Amamiya. The design and application of kodama system. *IEICE Transactions INF.& SYST.*, E85-D(04):637–646, 4 2002.