Multi-Agent Based Peer-to-Peer Information Retrieval Systems With Concurrent Search Sessions

Haizheng Zhang, and Victor Lesser
Dept. of Computer Science, University of Massachusetts Amherst, MA 01003
hzhang@cs.umass.edu, lesser@cs.umass.edu

ABSTRACT

In cooperative peer-to-peer information retrieval systems, each node can be considered an intelligent agent and these agents work collectively to provide an information retrieval service. In order to effectively support multiple and concurrent search sessions in the network, we propose two traffic engineering techniques that minimize processing and communication bottlenecks. One is a novel agent control mechanism whose elements include resource selection, local search scheduling, and feedback-based load control. The other is a new two-phase query routing algorithm based on organizational knowledge. Experimental results show that this framework can reduce congestion situations, increase system throughput, and improve considerably the overall system utility.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Algorithms

Keywords

peer-to-peer information retrieval, agents, control

1. INTRODUCTION

In recent years, peer-to-peer based information retrieval(IR) systems have begun to receive considerable attention. A peer-to-peer based information retrieval system consists of a set of nodes connected in a peer-to-peer fashion. Each node hosts a document collection to share with other nodes and these nodes work together to provide information retrieval service to users. In this paper, we consider an information retrieval task as a search session during which nodes forward queries to neighbors, perform IR operations and/or return search results. The lack of complete, up-to-date information about the states of other nodes in the network requires sophisticated coordination strategies. In addition, the presence of concurrent search sessions adds another level of complication due to bandwidth and processing capacity limitations: nodes may not be able to complete forwarding and local searches in a timely fashion for all queries they have received. Therefore, nodes have to decide what actions to take for each search session and in what order to execute those actions in order to maximize overall system utility. These local decisions, taking place at each agent, collectively determine the way concurrent queries are distributed and processed in the network.

In this paper, we extend a multi-agent based infrastructure [11, 12] developed for a single search session such that it can handle multiple, concurrent search sessions. These extensions required the development of a new agent control mechanism and a novel coodinated query routing algorithm based on a queueing theory analysis. Both mechanisms control the flow of information from different perspectives in the network to exploit available resources while reducing the occurrence of congestion situations. The purpose of the local control mechanism, based only on the agents' local observations of their neighboring agents, is to improve the average effective propagation speed of search sessions, a concept that will be introduced in later sections. The elements of such a control mechanism include resource selection, local search scheduling and feedback-based load control. The coordinated search algorithm is a modified version of a previously developed two-phase query routing algorithm [12] that exploits static agent organizational knowledge to locate relevant documents quickly while at the same time minimizing hot-spots caused by concurrent search sessions. Experimental results show that this framework can reduce the frequency of congestion situations, increase system throughput, and improve overall system quality.

The contributions of this work include: (1) a novel agent control mechanism based on a queueing theoretical analysis; (2) a simple feedback-based probabilistic load control component to avoid congestion situations while dynamically exploiting available communication and processing bandwidth; (3) a distributed, two-phase query routing algorithm based on a hierarchical structure of agents that balances search traffic in the network.

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. Fourth, we assume that the document collections hosted on agents are disjoint, and therefore the total number of relevant documents for a certain query is the sum of the relevant documents returned from each agent. 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 remainder of this paper is organized as follows: Section 2 discusses related research; Sections 3 and 4 introduce the system architecture of a P2P information retrieval system and the individual agent structure respectively. Section 5 presents the local scheduling algorithm in each agent. Section 6 introduces a two-phase search algorithm based on a hierarchical agent organization; Section 7 provides the experimental settings and explains the results; Section 8 discusses future work, and Section 9 summarizes the major contributions of this paper.

2. RELATED WORK

This paper is related to two lines of research: (1) query routing algorithms in peer-to-peer information retrieval systems, and (2) traffic engineering techniques in IP-Level and application-level network research.

The first line of relevant research includes recent studies on pure peer-to-peer information retrieval systems and hierarchical peer-to-peer systems [10, 7, 11, 12]. Lu proposed a framework including resource selection, resource representation and result merging in a hierarchical P2P environment[7]. In particular, a language model based search algorithm is employed in a top-down fashion. Indeed, as a query routing process can be considered as a distributed search process, it is natural to study this system in a multi-agent framework [5, 11, 12]. However, most of these research efforts do not consider situations with multiple, concurrent search sessions which is the focus of this paper.

The second line of relevant research is on network traffic engineering. Traffic engineering, which has mostly concerned IP-level routing problems, involves adapting the routing of traffic to network conditions, with the joint goals of good user performance and efficient use of network resources. The most well studied work along this line is on congestion control which attempts to minimize delays in Internet routing[3]. While query routing in P2P based information retrieval differs considerably from IP level routing problems, there are similarities that can be exploited in query routing algorithms.

3. PROBLEM DESCRIPTION

All agents in a peer-to-peer information retrieval system constitute a graph $G(A, E)$. Set A is the set of all agents in the system while set E includes all connections among agents. A search session s_i starts when an agent A_j receives a query qry_k from a user at time t_l. During the search process, upon receipt of the query qry_k, agents conduct local searches, forward the query to their neighbors and return search results to agent A_j. Agent A_j keeps a set of result tuples $T_m(s_i, A_m, t, Q_{s_i,A_m})$, with each element in the set specifying the fact that an agent contributes a certain amount of quality, Q_{s_i,A_m} in this case, to search session s_i at time t. The search conducted by agent A_m can be evaluated off-line by considering the ratio of the number of relevant documents returned from agent A_m to the total number of relevant documents in the network. Notice that the size of the result tuple set may keep growing with more search results returned from agents. The cumulative quality for a search session at time t is therefore the sum of the quality fields in the result set T_k obtained at time t. Formally, it is defined as:

$$Q_{s_i}(t) = \sum_{A_k} Q_{s_i,A_k} \text{where } T_k.t < t$$

In the search process, agents discard those queries they have previously processed thereby preventing queries from looping in the system forever. The search session ends when all the agents that receive the query drop it. We define the period from a query entering the system to its leaving the system as the duration of the search session, t^*. Therefore, the maximal quantity for a search session is defined as $Q_{s_i}(t^*)$. The duration of a search session depends on factors including the connectivity of the agent organization, the query routing algorithm and local scheduling algorithms in the agents. In general, a long search session duration indicates the occurrence of hot spots in the network. These hot spots are usually characterized by communication congestion or long query queue length in agents.

In this paper, we focus on open information retrieval systems, meaning that queries can enter the network at any agent from outside users, thereby forming multiple concurrent search sessions in the system. In our model, we define the arrival rate of jobs from outside to the ith node of the network to be λ_{0i}. The distribution of the incoming queries to a single agent conforms to a Poisson distribution, and the overall arrival rate λ from outside to an open network is:

$$\lambda = \sum_{i=1}^{N} \lambda_{0,i}$$

In the presence of multiple concurrent search sessions, we define the overall system utility, GQ, as the sum of the cumulative utility for all the search sessions including both the outstanding search sessions and finished search sessions under the assumption that all search sessions are considered equally important:

$$GQ(t) = \sum_{s_i} Q_{s_i}(t)$$

4. AGENT INTERNAL STRUCTURE

In this section, we describe the internal structure and local scheduling algorithms of each individual agent. Figure 1 illustrates the four components of our agent architecture: a document collection with its associated collection descriptor, a search engine, an agent view structure and a control unit. This section describes the first three components while Section 5 describes the control unit.

The document collection hosted on an agent includes a set of documents to share with other peers. The local search engine allows each agent to conduct local searches and return

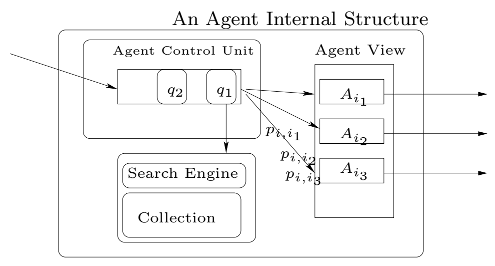

Figure 1: Internal Agent Structure

relevant documents upon receiving user requests. The collection can be described by a collection descriptor, which can be considered as the "signature" of the collection. By distributing collection descriptors, agents can gain knowledge about how content is distributed in the agent network. A collection descriptor characterizes the distribution of the vocabulary in the collection. It is based on the language model concept[9] that has proved effective in various previous studies in the distributed IR field [1, 7, 11]. A language model has many interesting properties which are easily exploitable in the peer-to-peer network system. Most importantly, it is a lightweight, concise and accurate descriptor of the document collection. Additionally, the size of the collection model is almost independent of the size of the document collection.

The agent view structure is an agent's restricted view of the network and determines the underlying topology of the agent "virtual" organization. Each agent's view structure contains the collection descriptors of a set of other agents' collections as well as the addresses of these agents and other related information such as the connection type. The design of the agent view structure depends on the underlying topological agent organization. In a power law based flat agent organization, only one connection type is needed as all agents play the same role. On the other hand, there can be multiple connection types in a hierarchical peer-to-peer agent organization, including links connecting upper level agents to lower level agents, lower level agents to upper level agents and agents at the same level. The specifics of the connection type information can be used in deciding how to route queries. More information about the agent view structure and the role it plays in search process is detailed in [11, 12].

5. AGENT LOCAL CONTROL MECHANISMS

The information retrieval performance, when there are concurrent search sessions, depends on how resources are allocated among these search sessions. The lack of centralized control of these concurrent searches can lead to an uneven distribution of query load among agents; this uneven load can then cause the situation where some agents are flooded with incoming queries while other agents are largely idle. Mitigating this problem requires agents to be able to control the message flow from a local perspective to prevent congestion and speed up the propagation speed of search sessions in the network. In addition to congestion prevention, in order to maximize overall system utility, agents should disseminate queries only to relevant agents so as to exploit bandwidth efficiently and reduce communication and processing load.

In this section, we present an analytical model based on queueing theory for agent control. In particular, the control unit of an agent makes decisions regarding which queries to process locally, in what order to process them (local search), and whether and to whom should the queries be forwarded (resource selection).

5.1 Resource Selection Algorithms

The resource selection algorithm chooses a subset of neighboring agents (that may contain relevant documents or have information about the location of relevant documents) to forward a query for further processing. Resource selection algorithms are characterized by their IR calculation overhead, the communication load they induce and how "focused" the search is. Existing resource selection algorithms include broadcast based routing algorithms, probabilistic based random routing algorithms and language model based algorithms[11, 12, 7]. Each resource selection algorithm incurs a certain amount of cost depending on its specific operations. For instance, a broadcast algorithm needs the least calculation, incurs minimal selection overhead but can easily saturate the network with messages, whereas a probabilistic random approach saves communication load but increases local computation load. A language model based approach has a different trade-off. It requires relatively expensive comparison operations but can significantly reduce the number of messages propagating in the system. Expensive computation operations can reduce agents' throughput which will be described in the next sections. Therefore, in the presence of concurrent search sessions, it is not always preferable to use a resource selection algorithm that generates minimal communication load but requires expensive comparison operations. The trade-off between message number and the amount of computation will be analyzed in the following sections.

5.2 A local scheduling mechanism

Each individual agent control unit maintains several queues including message queues, which contain messages to be forwarded to specific agents, and a local query queue, which contain queries waiting for further local processing. The set of agent control units in the network constitutes a queueing network system. In this section, two approaches for organizing the queueing network are introduced and compared based on their utilization rate and throughput.

Several assumptions are made in order to simplify the analysis.

(1) The service times at each queue including resource selection, local search scheduling and message forwarding are exponentially distributed.

(2) The inter-arrival times of queries are random and have Poisson distribution.

(3) The channel capacity between two arbitrary agents, A_i and A_j, is fixed, denoted as C_{ij}.

(4) The length of queries in the system, L, is considered a constant. This assumption simplifies the calculation of the throughput of communication channels.

Figure 2 shows an agent's control unit and indicates how resource selection, local searches and forwarding for parallel queries are related. This model consists of Q_{rs}, a queue that contains queries to be forwarded to other neighboring agents, Q_{ls}, a queue that contains queries waiting for further local search operations, and message forwarding queues Q_{1j} for each agent A_j in the agent view structure. These queues are considered as M/M/1 queues based on the above assumptions.

The arrival rate λ_i for queue Q_{rs} and Q_{ls} in agent A_i is calculated based on the departure rate, λ_{ji}, from its upstream agent A_j, and the arrival rate, λ_{0i}, from the outside users to agent A_i,

$$\lambda_i = \lambda_{0i} + \sum_{j=1}^{N} \lambda_{ji} \tag{1}$$

for i = 1, .., N

However, in a practical system, agents discard those queries that have been processed previously to ensure loopless communication. We assume the arrival rate of non-redundant queries is $\lambda_i^{'}$.

The resource selection time spent on a query is denoted as t_{rs} and the service time for local search is t_{ls}. Recall that both service times satisfy exponential distribution.

Queue Q_{rs} and Q_{ls} have the same arrival rate as specified in Equation 1. The throughput for Q_{rs} is $\frac{1}{t_{rs}}$ and for Q_{ls} is $\frac{1}{t_{ls}}$. Note that by separating the queue into two queues, the local search operation is no longer the bottleneck. Even when the incoming rate is higher than the service rate of Q_{ls}, an agent can buffer the incoming queries for later processing.

The arrival rate for each message queue Q_{ij}, λ_{ij} is the number of messages departing from Q_s in a time unit, i.e $\lambda_{ij} < \lambda_i^{'}$. The utilization rate for message queue Q_{ij} is $\rho_{Q_{ij}} = \frac{\lambda_{ij}}{\mu_{Q_{ij}}}$ And the throughput for Q_{ij} is

$$\mu_{Q_{ij}} = C_{ij}/L \tag{2}$$

.

Because each agent contributes to the overall system utility for each finished local search, the performance of the scheduling algorithm for the local search queue determines how fast the overall utility is accumulated. Moreover, because the selection of one query is always at the expense of executing other search sessions at a later time, this algorithm also affects the fairness of the network search process. It is worth mentioning that local fairness does not necessarily lead to global fairness.

A simple scheduling algorithm for the local query queue is first come first serve(FCFS) where agents always process queries that enter the system earliest. Although the FCFS algorithm is fair from an agent's local perspective, it can cause significant delay for queries that enter the system at a later time. An alternative scheduling approach is based on a greedy algorithm that aims to maximize the overall utility from an agent's local perspective.

$$GQ(t) = \sum_{s_i} Q_{s_i}(t) = \sum_{A_i} Q_{A_i}(t)$$

$Q_{A_i}(t) = \sum_{s_i} Q_{s_i,A_i}$ is defined as the local accrued utility of agent A_i. Therefore, in order to maximize the overall utility, each agent selects the most relevant subset of queries to process. In order to maximize the local utility, we use a heuristic based sorting algorithm to order the queries queue according to the Kullback-Leibler (KL) distance between the query model and the local collection model. KL distance is a metric to evaluate the distance of two distributions. The calculation detail is introduced in [12].

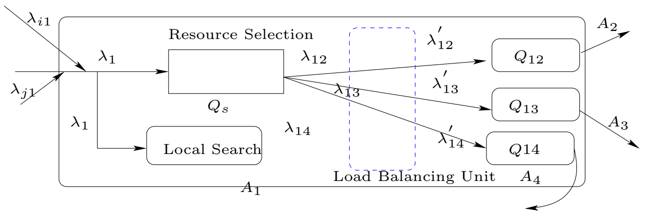

Figure 2: The agent internal structure with a parallel control mechanism

5.3 A Feedback-based, Probabilistic Load Control Unit

The above section analyzed the behavior when the queueing network is stable. However, when the query arrival rate is more than the service rate, the queue length will be ever expanding. In this section, we first analyze the cause of "hot spots" in the above model and then present a feedback based probabilistic load control mechanism in order to avoid or mitigate congestion situations.

5.3.1 *The creation of "Hot Spots"*

In the above model, there are two cases that would create a congestion situation: (1)The arrival rate for message queues (Q_{12}, Q_{13}, etc) is above the service rate, i.e

$$\lambda_{ij}^{'} > \frac{C_{ij}}{L}$$

(2)The arrival rate for queue Q_{rs} is more than the service rate, i.e:

$$\lambda_i > \frac{1}{t_{rs}}$$

From equation 1, it is equivalent to:

$$\lambda_i = \lambda_{0,i} + \sum_{j=1}^{N} \lambda_{ji} > \frac{1}{t_{rs}}$$

In the following section, we propose a load control mechanism to avoid both situations, thereby improving overall system performance.

5.3.2 *A Probabilistic Load Control Algorithm*

In order to control the communication load based on limited observations of the network, agents can employ a feedback-based, probabilistic algorithm. Specifically, when an agent forwards queries to its neighboring agents, it not only considers the capacity of its own communication channels, but also takes into account its neighboring agents' service rates, which are acquired dynamically by analyzing the feedback information sent out by its neighboring agents periodically.

Let QS_{ij} be the query set that the resource selection algorithm of agent A_i selects to forward to agent A_j and QS_i

be the query set that agent A_i receives and has not processed previously in a certain time period t. Assume that in a certain period, queue Q_s receives non-redundant query set QS_i and the load control mechanism randomly selects $p_{ij} * |QS_{ij}|$ from the query set QS_{ij} and $q_{ij} * |QS_i - QS_{ij}|$ from the set $QS_i - QS_{ij}$. Notice that $|QS_{ij}| = \lambda_i^{'} * t$ and $|QS_{ij}| = \lambda_{ij} * t$

Agent A_i determines probability p_{ij} and q_{ij} in the following way:

(1) $q_{ij} >= 0$ only if $p_{ij} = 1$.

(2) $p_{ij} = min(p_1, p_2, 1)$ where

$$p_1 = \frac{\frac{C_{ij}}{L}}{|QS_{ij}|}$$

and

$$p_2 = \frac{f_{ij}}{|QS_{ij}|}$$

(3) q_{ij} is determined by:

$$q_{ij} = \frac{min(f_{ij}, \frac{C_{ij}}{L}) * t - |QS_{ij}|}{|QS_i - QS_{ij}|}$$

(4) f_{ij} is determined by agent A_j and forwarded to agent A_i as feedback to control the number of messages to be forwarded to agent A_j from agent A_i. In our work, f_{ij} is determined by

$$f_{ij} = t_{ij} * \lambda_{ij} * t$$

and

$$t_{ij} = \frac{\frac{1}{t_{rs}} - \lambda_{0j}}{\lambda_j - \lambda_{0j}}$$

Recall that λ_{0i} is the arrival rate of queries from outside users instead of other agents. t_{ij} characterizes the overload situation at agent A_j

(1) specifies that agents selected by the resource selection algorithm have higher priority over other agents. Agent A_i would only forward a query to more agents when the utilization rate of communication channel is lower than 1.

(2) prevents the situation where too many queries are scheduled for the communication channel between A_i and A_j. In particular, p_1 is the maximal capacity of the communication channel and p_2 is the maximal number of queries that agent A_j can process in a timely fashion for agent A_i.

(3) decides that when the query arrival rate is low, the load control unit should increase the utilization of the communication channel and attempt to increase the speed that search sessions propagate in the system by forwarding more queries to agent A_j than the set that was selected by the resource selection algorithm.

(4) describes how the feedback information is generated. Agent A_j requests the upstream agents to reduce communication load by t_{ij} in order to prevent congestion situations.

6. A BALANCED TWO-PHASE QUERY ROUTING ALGORITHM

The control mechanism described above engineers traffic flow in a distributed information retrieval system from a local perspective. The effect of this mechanism, however, could be limited by the fact that agents only possess a narrowly defined non-local view of content network traffic. In this section, we introduce a query routing algorithm which exploits a more encompassing but static view of the network topology to route queries. It is the interaction among these two mechanisms that is key to our approach.

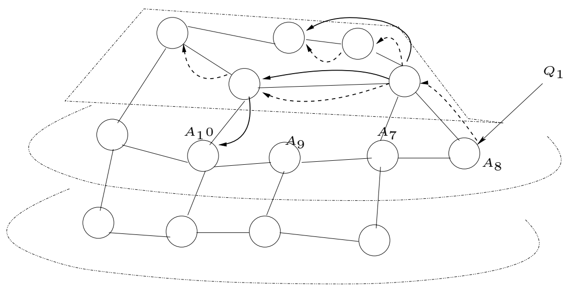

Figure 3: A Two-Phase Query Routing Algorithm In Hierarchical P2P IR System

In our previous work[12], we proposed a hierarchical agent topology and a two-phase search algorithm for use in routing queries. In this section, we extend this work to make it suitable for concurrent query processing in the network. In our hierarchical agent society, agents have two roles: group-mediator and query-processor[12]. 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.

To take advantage of hierarchical agent organization, a two-phase hierarchical search algorithm was developed in [12]. In this protocol, as illustrated in Figure 3, agents in the first phase forward queries to top-level mediators so as to find appropriate starting points for the search. In the second phase, a concurrent search is initiated from these starting points. The intention of this algorithm was to conduct searches in a focused fashion to reduce communication messages. Notice that the load may become unbalanced in such a network. The arrival rate for a top-level mediator A_i is:

$$\lambda_i = \lambda_{0,i} + \sum_{j \in \{A_i^{'}\}} \lambda_{0,j} + \sum_{j=1}^{N} \lambda_j p_{ji}$$

for i = 1, .., N. Here we denote $\{A_i^{'}\}$ as the set of all direct or indirect members of agent A_i. In the above equation, $\lambda_{0,i}$ is the query arrival rate from users, while $\sum_{j \in \{A_i^{'}\}} \lambda_{0,j}$ is the query arrival rate coming from the member set of agent A_i; $\sum_{j=1}^{N} \lambda_j p_{ji}$ is the query arrival rate coming from the neighbors set. This makes the resource selection algorithms very expensive since the selection algorithm at top-level mediators must perform comparisons of query model and collection model of each agent. This algorithm is not a balanced approach since it tends to increase dramatically the burden of top-level mediators, thereby creating hot spots in the network. To resolve this problem, we propose an improved two-phase search algorithm as demonstrated in Figure 4.

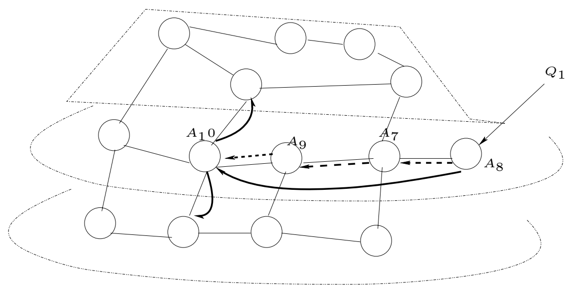

Figure 4: A Balanced Two-Phase Query Routing Algorithm

The first phase of this query routing algorithm is primarily conducted in the horizonal direction. It starts when the initiator of the query, A_8, forwards the query similarity probe messages with a certain TTL (Time To Live) value along the lateral links to the agents at the same level to locate relevant clusters. Upon receipt of these probe messages, agents return back the similarity values of their collections with the queries in question. After the TTL value expires, an agent does not forward the query any further, thus stopping further search along this path. After comparing the similarity returned by the agents, the initiator, $A8$, selects the K most similar agents to proceed to the second phase of the search. The second search phase is primarily conducted inside each group and information flows in the vertical direction. In particular, agents only forward the query along upward links or downward links during the second phase. For evaluation purposes, there is no explicit recognition by individual agents that the query is no longer being processed by any agent in the network. 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. In reality, this phase can also be controlled by a TTL value to limit search efforts. This way, the traffic load is likely to be more evenly distributed among the various levels of the hierarchy given that the entry points of the queries are randomly distributed in the network. Hence, the new search strategy mitigates the hot spot problem encountered in the previous algorithm.

7. EXPERIMENTAL SETTINGS AND RESULTS ANALYSIS

7.1 TRANO Testbed

TRANO(Task Routing on Agent Network Organization) is a multi-agent based information retrieval testbed. TRANO is built on top of the Farm simulator[6] that provides a data dissemination framework for large scale distributed multi-agent organizations. TRANO supports importation and exportation of agent organization profiles including topological connections and other features. Each TRANO agent is composed of an agent view structure and a control unit. In simulation, each agent is pulsed regularly and the agent checks the incoming message queues, performs local operations and then forwards messages to other agents .

7.2 Experimental Settings and Results Analysis

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 originally split from TREC VLC1 collection by data sources in order to create testbed for distributed information retrieval research[2]. TREC VLC1 is part of the TREC collections which are distributed by the National Institute of Standards and Technology (NIST) for testing and comparing current text retrieval techniques. TREC VLC1 (very large collection) includes documents from 18 different data sources, such as news, patents, and the Web[4]. The formation of hierarchical agent organization can be achieved by a hierarchical clustering algorithm mentioned in [7]. However, this is under the assumption that global information about the content distribution is available during the topology formation. In the absence of global information, [12] proposed a distributed hierarchical clustering algorithm to form a hierarchical organization which is exploited in this paper. During the topology generation process, degree information of each agent is estimated by the algorithm introduced in [8] with parameters $\alpha = 0.5$ and $\beta = 0.6$. In our experiments, we estimate the upward limit and downward degree limit using linear discount factors 0.5, 0.8 and 1.0. Once the topology is built, queries randomly selected from the query set $301 - 350$ on TREC-VLC-921 are injected to the system based on a Poisson distribution

$$P(N(t) = n) = \frac{(\lambda t)^n}{n!} e^{-\lambda}$$

In addition, we assume that all agents have an equal chance of getting queries from the environment, i.e, λ is the same for every agent. In our experiments, λ is set as 0.00543 so that the mean of the incoming queries from the environment to the agent network is 5 per time unit. The service time for communication queue Q_s and Q_{rs}, i.e $t_{Q_{ij}}$ and t_{rs}, is set as 0.2 time unit. The service time for a local search queue, t_{ls} is 3 time units.

We use KL divergence to measure the distance among collection models or between collection models and query models. The formula is:

$$D(p||q) = \sum_i p(i) \log \frac{p(i)}{q(i)}$$

An approximation approach is used to speed up the calculation of KL distance and convert it to a similarity measure. The calculation details can be found in [11].

7.3 Performance Measures

The following measures were used to compare the different search strategies:

Propagation Speed of a search session s_i, $ps_{s_i}(t)$ is defined as the ratio of the number of agents visited at time t, $N(t)$, over the size of the agent society N, i.e $ps_{s_i}(t) = \frac{N(t)}{N}$. Average propagation speed is defined as average of the propagation speed of all the search sessions in the network. Notice that t is the amount of time after a search session starts. The maximal proportion of agents that a message visits is $ps_{s_i}(t^*)$.

Effective Propagation Speed of a search session s_i, $eps_{s_i}(t)$ is defined as the ratio of the number of relevant documents hosted on visited agents at time t,

Table 1: Experimental Results For Various Search Strategies

Name	Topology	RS	LC	λ	t^*	$ps_{s_i}(t^*) * t^*$	$eps_{s_i}(t^*) * t^*$	GU	MsgNumber	$GQ(t^*)$
PBN	PowerLaw	Broadcast	No	0.00543	32	97%	97%	92%	2603022	77.29
PBY	PowerLaw	Broadcast	Yes	0.00543	31	97%	96%	94%	2563046	78.99
HTN	Hierarchical	TwoPhase	No	0.00543	17	29%	33.5%	44%	353446	38.01
HTY	Hierarchical	TwoPhase	Yes	0.00543	16	32%	35%	50%	407242	45.20
HBN	Hierarchical	Balanced	No	0.00543	26	89%	92%	80%	1512011	83.94
HBY	Hierarchical	Balanced	Yes	0.00543	24	89%	98%	85%	1552011	88.94

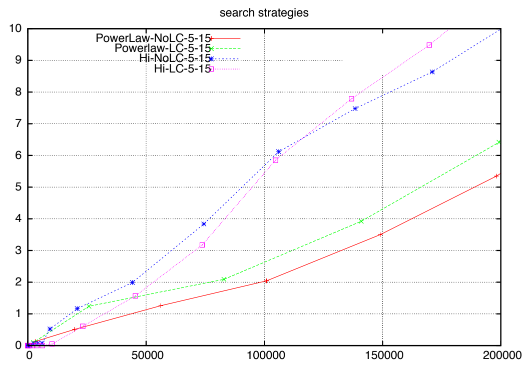

Figure 5: $GQ(t)$ versus the message number

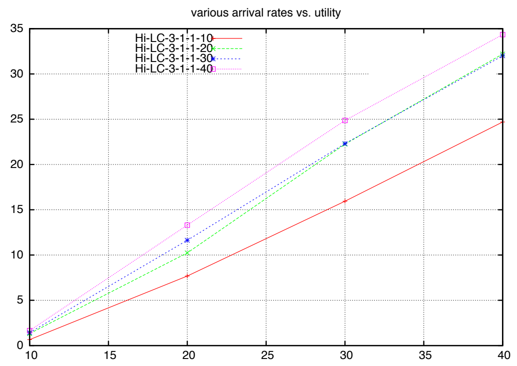

Figure 6: utility increase overtime for various arrival rates

$R(t)$, over the total number of relevant documents in the network R, i.e $eps_{s_i}(t) = \frac{R(t)}{R}$. Note that this ratio is not equal to recall ratio as the local searches may not be scheduled or finished by time t. Average effective propagation speed is defined as the average of the effective propagation speed of all the search sessions in the network. The maximal proportion of agents that a message visits in session s_i is $eps_{s_i}(t^*)$.

Global Utilization Ratio (GU) is defined as the average of the utilization rates of agents in the network. The utilization ratio of an agent is defined as the probability that the agent is busy and its queue is nonempty. A low global utilization ratio can be attributed to either a low query arrival rate or a poorly designed resource selection process and an agent search topology that creates unbalanced data flow in the network, leaving some agents with no queue entries while flooding others.

7.4 Results analysis and evaluation

Table 1 lists the experimental results of 6 different search strategies for 60 time units. These strategies explore the importance of using different topological agent organizations (powerlaw topology and hierarchical topology), search strategies (broadcast, two-phase hierarchical, and a balanced hierarchical search strategy). Fig. 5 depicts the global quality of the system as a function of the number of messages transmitted. *Hi-LC-5-15* is the curve for hierarchical, balanced two-phase search appraoch with local balancing techniques and *Powerlaw-LC-5-15* shows the performance for the broadcast search algorithm running on the powerlaw topology. In the following paragraphs, we analyze these experimental results and results shown in Fig. 6 that indicate how one specific search strategy is affected by the different arrival rates of the external queries.

Whether the load balance mechanism was used or not, the overall results were that the balanced hierarchical(details in Table 1) achieved highest overall utility while limiting the amount of communication relative to the broadcast approach. Additionally, the following findings were also revealed by the experimental results:

First, while the two-phase search algorithm was designed to take advantage of the hierarchical agent organizations, the way queries are forwarded in the system makes it easy to form hot spots that slow down the propagation of queries in the network. Therefore, from Table 1, the two-phase search algorithm produces low search quality. On the other hand, a balanced, hierarchical algorithm taking advantage of lateral links achieves the best search quality among the tested approaches in terms of both the number of communication messages and the overall utility. This is due to the fact that the queries are largely forwarded inside the relevant groups rather than being immediately forwarded to top level mediators in the two-phase search. This avoids the top-level mediators hot spots.

Secondly, a load balancing mechanism tends to improve system performance by reducing the number of messages when query arrival rate increases and congestion situations occur while increasing the number of messages in the network when the service load and utilization of the communication channel is low. From the experimental results, this load balancing mechanism improves the two-phase search algorithm the most. This can be attributed to the fact that the routing in two-phase search algorithm is very unbalanced with the query arrival rate used in these experiments. The preliminary experimental results also demonstrate that the contribution of load balancing depends on the local computational capacity. Particularly, the traffic engineering techniques contribute to the performance more when local searching is not a bottleneck. When the local search queue size increases to a certain level, optimizing the traffic does not significantly improve search performance. It remains as future work to further discover the correlation between traffic flow and global search performance.

Thirdly, Fig. 6 shows that the overall system performance increases steadily with the increase of query arrival rates. This demonstrates that the system is a stable one.

Finally, the broadcast approach, to our surprise, performs quite well. It generates a lot of extraneous messages which are then filtered out locally by the language model based greedy local search scheduling algorithm. Therefore, this large number of messages did not distract the agents' local searches.

8. DISCUSSION AND FUTURE WORK

One of the directions that we would like to pursue is to provide differentiated service for search sessions by using selectively different search engines and parameters that trade off the likelihood of finding relevant documents to the time required for the search. We also want to be able to specify that some search sessions are more important than others, and expand the model so that the utility function of each search session may not necessarily be linear to the quality accrued over time. Rather, we assume that the utility functions increase faster at the beginning and the increase slows down when a certain amount of quality has been accrued.

$Q_{s_i}(t)$ is an objective measure which reflects the recall ratio at time t for search session s_i. Another measure, user utility, is also defined to capture the preferences that users have on the number of relevant documents returned at different periods during the search process.

$$U_{s_i}(t) = \mu(Q_{s_i}(t), t)$$

The overall system utility, $GU(t)$ is defined as the sum of the utility for each search session in a certain time period:

$$GU(t) = \sum_{s_i} \mu(Q_{s_i}(t), t)$$

By selecting an appropriate μ function, users can specify their preferences for search session s_i, i.e whether they prefer quick response but with a relative low recall ratio or vice versa. In this paper, we take $\mu(x) = x$ and therefore we consider that user satisfaction linearly increases as more relevant documents are returned.

9. CONCLUSIONS

In cooperative peer-to-peer information retrieval systems, each node can be considered as an intelligent agent and these agents work collectively to provide an information retrieval service. In order to effectively support multiple and concurrent search sessions in the network, we propose two traffic engineering techniques that minimize processing and communication bottlenecks. One is a novel agent control mechanism whose elements include resource selection, local search scheduling, and feedback-based load control and the other is a new two-phase query routing algorithm based on organizational knowledge. Experimental results show that this framework can reduce congestion situations, increase system throughput, and improve considerably the overall system utility.

10. REFERENCES

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

[2] J. C. French, A. L. Powell, J. P. Callan, C. L. Viles, T. Emmitt, K. J. Prey, and Y. Mou. Comparing the performance of database selection algorithms. In *Research and Development in Information Retrieval*, pages 238–245, 1999.

[3] R. Gallager. A minimum delay routing algorithm using distributed computation. In *IEEE transactions on communications*, pages 73–85, Jan 1977.

[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] B. Horling. *Modeling and Designing Multi-Agent Systems Through Explicit Organizational Design.* PhD thesis, University of Massachusetts at Amherst, Amherst, Massachusetts, 2005.

[6] R. Horling, Bryan; Mailler and V. Lesser. Farm: A scalable environment for multi-agent development and evaluation.

[7] J. Lu and J. Callan. Federated search of text-based digital libraries in hierarchical peer-to-peer networks. In *In Proceedings of the Twenty-Seventh European Conference on Information Retrieval Research (ECIR'05).*

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

[9] 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.

[10] M. E. Renda and J. Callan. The robustness of content-based search in hierarchical peer to peer networks. In *CIKM '04: Proceedings of the thirteenth ACM conference on Information and knowledge management*, pages 562–570, New York, NY, USA, 2004. ACM Press.

[11] H. Zhang, W. B. 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.

[12] H. Zhang and V. R. Lesser. A dynamically formed hierarchical agent organization for a distributed content sharing system. In *2004 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2004), 20-24 September 2004, Beijing, China*, pages 169–175. IEEE Computer Society, 2004.