Adaptive Routing Algorithms in Unstructured Peer-to-Peer (P2P) Systems

Achmad Nizar Hidayanto
Faculty of Computer Science
Universitas Indonesia
Jakarta, Indonesia

Stephane Bressan
School of Computing
National University of Singapore
Singapore

Abstract—There are a variety of peer-to-peer (P2P) systems for sharing documents currently available. According to their data organization, P2P systems are classified into two categories: structured and unstructured P2P systems. In structured P2P systems, peers are organized according to some mapping techniques, e.g. hashing function. Whereas in unstructured P2P system, peers are connected to each others randomly; resources are not moved to other peers but hosted on site. Unstructured P2P systems offer a more flexible and autonomous environment, as they require less control for the placement of resources and peers. This work focuses on experimenting on unstructured P2P systems. The challenge in unstructured P2P system is designing routing strategies that lead the user in finding the documents needed. Routing strategies in unstructured P2P system need to consider the dynamic aspects of P2P systems; peers are dynamic and constantly joining and leaving the system, network load changes continuously and resources are added and removed over the time. Therefore, the routing strategy must adapt to such changes to maintain its performance. We propose routing strategies that adapt to these changes through learning mechanisms. The learning mechanisms are conducted by observing the internal and external behaviors of the system. Internal behaviors reflect the internal state of peers such as peers' interest and collection. External behaviors reflect the external state of the system such as network load. In order to measure the performance of the proposed routing algorithms, some common performance measurements are used. These are "response time" and "number of messages generated" or what is commonly referred to as efficiency, "number of answered and satisfied queries" and the "similarity of documents" or what is commonly referred to as effectiveness of retrieval system. The experiment results show that the proposed algorithms are capable of adapting to new changes. By learning to adapt, the system maintains its performance in terms of efficiency and effectiveness. Moreover, comparison with other similar algorithms also shows the superiority of the proposed routing algorithms. Thus, the proposed routing algorithms are good candidates for effective and efficient retrieval of documents in P2P systems.

Keywords- Adaptive routing algorithms; peer-to-peer system; document retrieval; routing index; interest groups; expert groups; hybrid groups;

I. INTRODUCTION

Peer-to-peer (P2P) systems have recently attracted a lot of attention since it allows the implementation of large distributed repositories of digital information. The digital information can be in the form of video, audio, image, html, text file, etc. P2P systems consist of thousands, even millions, of peers sharing their resources in equal roles. Each peer provides services to other peers by sharing their resources; these peers can also get resources from other peers. Therefore the main problem in P2P systems is how to locate resources that are scattered in the network efficiently and effectively. Most of researchers measure such performance in term of hop, where one hop refers to a trip of message from a peer to another one without any intermediate peers.

Nowadays we can see many emerging P2P applications. Most of them are used for file sharing systems. Many of those applications adopt and adapt previously established protocols such as Napster [1], Gnutella [2],

FastTrack [3], Freenet [4] and so forth. According to online Wikipedia [5], there are more than one hundred applications built on top of P2P technologies.

In terms of network topology, many P2P systems follow a semi-centralized or hybrid [6] architecture (e.g., Napster [1]), where queries are posed to a centralized index whilst data and services are distributed. Each peer registers its resources to the index. Despite their advantages, hybrid systems inherit the drawbacks of centralized architectures as they still cause single points of failure when querying from the index.

As an alternative, several fully distributed (or pure P2P) systems have been proposed. In these fully decentralized systems there are no centralized catalogues or functionalities; instead, peers are individually contacted and return the results they contain. According to their data organization, P2P systems can be divided into two main classes: structured and unstructured P2P systems. In structured P2P systems (e.g., Chord [7], CAN [8] and Past [9]), peers are connected to each other and resources or the records are moved to a peer according to some mapping techniques. Although they guarantee the location of content within a bounded number of hops, they require tight control of data placement and the network topology. In unstructured P2P systems (e.g., Gnutella [2], Freenet [4]), peers are connected to each other as they join the network and resources are not moved to other peers but are hosted on site. Hence, searching for a resource requires either broadcasting the request or using routing information.

Thus the challenging problem in unstructured P2P systems is designing a routing strategy that enables the user to find the documents needed. Routing strategies in unstructured P2P systems need to consider the dynamic aspects of P2P systems as the peers constantly join and leave the system, network load changes continuously and resources are added and removed over the time. Therefore, the routing strategy must be able to adapt to such changes in order to maintain its performance.

This research studies routing strategies that can adapt to the changes through learning processes. The algorithms are mainly constructed based on [10], [11], [12], [13], [14]. The learning processes are conducted by observing the internal and external behaviors of the system. The internal behaviors reflect the internal state of peers such as the peers' interest and collection. The external behaviors reflect the external states of the system such as network load and peers' load. By adopting the concept of propagation, the changes are propagated to the whole system to reflect the most recent states of the peers in the network.

The proposed routing algorithms also consider the retrieval technique as in documents retrieval system. Many previously developed P2P systems such as Napster, Chord, and Kazaa support only documents retrieval based on their identities. Incorporating the retrieval technique in those systems will provide comfortable and easier way for the users to express their information needs.

II. RELATED WORKS

Researchers commonly categorize P2P systems according to level of decentralization. According to Karl Aberer et al [15], P2P systems can be classified into three, namely: centralized, decentralized, and hierarchical (hybrid) P2P systems. In the centralized model, the system has a global index held by a central authority. Peers request file sending their queries to the central server. After receiving responses from the server, the requestor makes direct contact with the peer providing the resource. The example of P2P system that implements such a scheme is Napster. In the decentralized model, there is no global index and central coordinator. Global behaviors emerge from local interaction among peers. Examples of this kind of P2P systems are Gnutella, Freenet, etc. In the hierarchical model, the system introduces the concept of super-peers. A super-peer knows everything about all peers connected to it. This model is a mix between the centralized and decentralized models. Examples of P2P systems which use this approach are FastTrack and Kazaa.

According to peer organization, decentralized P2P systems can be divided into two main classes namely: structured and unstructured P2P systems. In structured P2P systems such as Chord [7], CAN [8], and Past [9], peers are connected to each other and resources are moved to a peer according to some mapping techniques. For example, Chord employs a hashing technique to map resources to a key and peers to a key range. Peers host resources whose keys fall within their key range. In unstructured P2P system architecture as in Gnutella [2] and Freenet [4], peers are connected to each other randomly as they join the network and resources are remaining hosted on site. Hence, searching for a resource requires either broadcasting the request or using routing information.

Some improvements had been proposed to increase the performance of routing in unstructured P2P systems. Beverly Yang and Hector Garcia-Molina [6] proposed three techniques to broadcast messages from a peer to its neighbors, two of which are put in this category: Iterative Deepening and Local Indices. In this research, Arturo Crespo and Hector Garcia-Molina [16] introduced the concept of Routing Indices (RIs). RIs allow peers to forward queries to a neighbor that is more likely to have answers. If a peer cannot answer a query, it will forward

the query to its best neighbor, based on its local RI, instead of selecting its neighbors randomly or flooding the network by forwarding the query to all its neighbors. The proposed routing indices named: Compound RI, Hop-count RI and Exponentially Aggregated RI.

Instead of improving the performance by creating more robust routing algorithms, the researchers also focus the improvement by creating overlay networks and replicating the resources. The first approach is the design of an overlay network based on the distance between a node and its neighbors [17]. In this approach, the overlay is formed by creating two types of links: short-link and long-link. Other approaches are by clustering peers based on their content/interest similarity. Among others are the studies by Vazirgiannis et al [18], Cheuk Hang Ng et al [19], Ramaswamy et al [20], Kalogeraki et al [21].

III. PROPOSED DESIGN

The proposed adaptive routing algorithms are categorized into two classes: routing algorithms that are adaptive to internal behaviors (internal states of the peers) and routing algorithms that are adaptive to external behaviors (external states of the peers). Both strategies rely on the routing indices to locate the answers. To continuously learn the internal and external states, the proposed systems use propagation of routing indices. Figure 1 shows the whole components of the proposed designs.

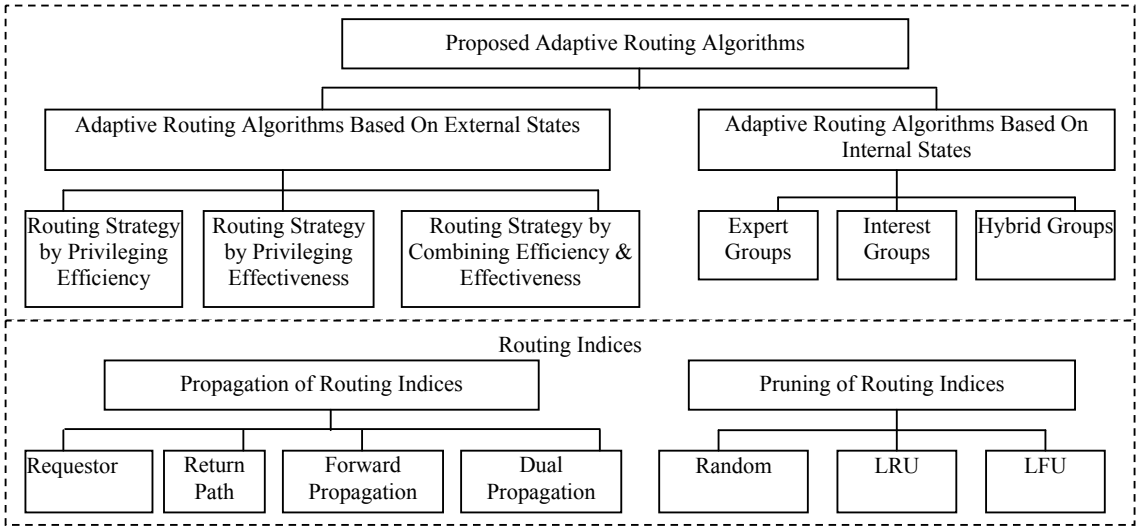

Figure 1. Research framework

The following section explains the general routing strategy in unstructured P2P systems. Second section explains the proposed adaptive routing that can adapt to external states. Third section explains the proposed adaptive routing that can adapt to internal states. That section describes how to exploit expertise/interest of the peers for creating groups that can be used to help routing process in P2P systems. Finally, the last section explains some proposed pruning of routing indices mechanisms. The routing indices contain information that can be used to help the routing process.

A. General Routing Processes in Unstructured P2P Systems

In P2P systems, peers store collections of documents. Queries are lists of terms or keywords and the retrieval of documents uses a similarity measure like cosine similarity. The network of peers is unstructured. Each peer maintains a routing index for the routing of queries. A routing index introduced by Crespo [16] is a data structure that records a list of potential relevant neighbors for each topic in a selected set of topics or keywords. The maximum number of topics and the maximum number of neighbors are parameters of the system that can be set explicitly or managed indirectly with an appropriate replacement strategy such as Least Recently Used (LRU) and Least Frequently Used (LFU).

A general form of query is $q(\tau, \lambda, \eta)$ where τ is a (possibly weighted) list of keywords. It forms a vector represents a topic. λ is the time-to-live (TTL), i.e. global expiry time or maximum number of hops allowed for the query (this is to avoid ghost queries from indefinitely circulating in the network). η is the maximum number of

documents requested. The form of query can be extended for various purposes, for example by adding parameter θ as a threshold for minimum similarity of retrieved documents or by parameter w to control the user preference.

A peer receiving the query compares the documents it stores with the query. The peer computes the similarity between τ and the documents represented by weighted vector of keywords or terms in the vector space model. If the similarity is above the threshold θ, the document is returned as an answer of the query. If the time-to-live is not yet reached (i.e. λ is not equal to 0, λ is number of hops), the query is forwarded to some neighbors (and λ is decremented if λ is number of hops). The neighbors can be selected randomly or using a heuristic function.

After receiving all documents from answering peers, the peer issuing the query will choose top η documents that have the highest similarity.

B. Adaptive Routing Strategies Based On External States

External states refer to the environmental condition of the peers such as network load and documents hosted on other peers. This research proposes to use adaptive routing strategies for the routing of requests, which are adaptive to such conditions. Those routing strategies are based on reinforcement learning: the Q-Routing. Q-Routing is an adaptive routing strategy that uses Q-learning, a reinforcement learning algorithm. Q-Routing has been studied in various network routing and literature and the performance studies show it is able of adapting to network traffic. This research adapts the mechanism in Q-Routing [22] and implements this adaptation for document searching. In original Q-Routing, the routing indices contain information stored about estimations of routing time to particular address through each neighbor. In the proposed algorithms, the routing indices contain information about either estimation of routing time or similarity to particular topic through each neighbor.

Each peer maintains a routing index, which consists of information about the status of the network in the local view of the peer. This section explains three proposed approaches and corresponding component. First subsection explains the design of routing indices for this purpose. Second subsection explains how to learn the network status to locate documents efficiently. Third subsection explains how to learn the collection owned in other peers to locate the documents that have high similarity to the query. Finally, the last subsection explains the algorithm combining efficiency and effectiveness to give users more flexibility in controlling queries.

1) Routing Indices Design

An important aspect to be considered to route queries efficiently and effectively is the design of the routing indices. The routing indices help peers selecting the best neighbor(s) to forward queries to. The first proposed approach adapts table of Q-values in Q-Routing to be implemented in the proposed adaptive routing algorithms. The design of the routing indices is similar to that of the indices for packet routing in network except that the entries containing IP addresses destination are replaced with the topics existing in the network.

The proposed adaptive routing algorithms should be capable of adapting to the changes of external states which are network load and collection of other peers. Therefore, the routing index in each peer should reflect the changes of the two aspects. In case of adaptive routing algorithm privileging efficiency the routing index maintained by a peer o contains, for each neighbor n of the peer and for each topic t, values denoted $T_o(n, t)$ (we call them T-values) which represent the estimated minimum time to retrieve at least a document similar to t by forwarding a query to n. In case of adaptive routing algorithm privileging effectiveness the routing index of each peer o maintains, for each neighbor n of the peer and for each topic t, values denoted $R_o(n, t)$ (we call them R-values) which represents the estimated maximum similarity to t of at least a document obtained by forwarding a query to n. In adaptive routing strategy combining efficiency and effectiveness the routing index of each peer o maintains, for each neighbor n of the peer and for each topic t, T-values and R-values. For the sake of simplicity, the form of routing indices that will be explained in this section is for adaptive routing algorithm combining efficiency and effectiveness. The form of routing indices for other two approaches can be inherited from them by only taking the respective column.

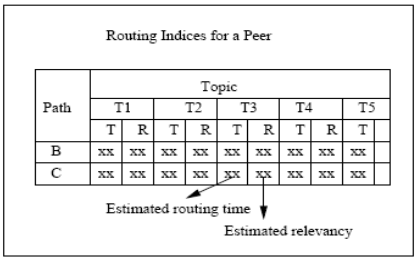

Figure 2. A routing index structure for adaptive routing algorithms based on external changes

Peers can seamlessly join and leave the network without having to tell the whole networks. The peers joining the network can initialize the content of the routing indices to the default and initial values, which is 0. Peers leaving the network can inform their direct neighbors to delete the corresponding entries in the routing indices. If a peer leaves the network on a failure, i.e. without informing its neighbors, its absence will be eventually discovered from its estimated performance in the routing indices.

For a peer *o*, each entry in the routing index is of the form:

$$(n, (T_o(n, t_1), R_o(n, t_1)), \ldots, (T_o(n, t_m), R_o(n, t_m)))$$

for each direct neighbor peer *n*, where *m* is the number of topics. Figure 2 shows the illustration of the form of routing index.

Values in the routing indices, i.e. T- and R-values, are updated upon sending or forwarding or receiving a query.

T-values for a peer o and a topic t are updated according to the following Q-routing formula:

$$T_o(n, t)^{new} = T_o(n, t)^{old} + l(T_n(t) + q_{o,n} - T_o(n, t)^{old}) \quad (1)$$

where *n* is the neighbor of *o* transmitting its best T-value for *t*, namely:

$$T_n(t) = \min_{p \in neighbour(n) \wedge p \neq o} (T_n(p, t)) \quad (2)$$

$q_{o,n}$ is the overhead communication cost from *o* to *n*, *l* is a number between zero and one which determines the learning rate. The bigger *l* is, the more sensitive it is to changes in the system. When *l* is set to 1, the equation becomes:

$$T_o(n, t)^{new} = T_n(t) + q_{o,n} \quad (3)$$

which updates the T-value without considering the old value.

R-values for a peer *o* and a topic *t* are updated according to the following formula:

$$R_o(n, t)^{new} = R_n(t) \quad (4)$$

where *n* is the neighbor of *o* transmitting its best R-value for *t*, namely:

$$R_n(t) = \max(\max_{p \in neighbour(n) \wedge p \neq o} (R_n(p, t)), \max_{d \in doc(n)} (rf(d, t))) \quad (5)$$

It is a learning process with a rate of 1 and an overhead cost of zero. The relevance function *rf* is used to compute the actual relevance (retrieval status value) of stored documents.

Clearly the estimated R-values are expected to be less subject to fluctuation than the T-values. Indeed, although both values depend on the network structure (peers leaving and joining), the T-value depends on the traffic (requests and responses) while the R-value depends on the documents' content and location. It is expected that the traffic to be the most dynamic element of the system.

The design of the routing indices accommodates changes in network load and the collection of other peers. However, the system can use it separately by removing one of information from the routing indices; either using T-values only or using R-values only in the routing indices. When the system uses information from T-values only, the system emphasizes the routing on efficiency as it only can predict the path that leads to the solution as fast as possible. When the system uses information from R-values only, the system emphasizes the routing on effectiveness as it only can predict the path that leads to the solution that has the highest similarity to the query.

The following subsections explain how to use information from routing indices to get different ways of locating the solutions that yield the three proposed adaptive routing algorithms which are routing strategy by privileging efficiency, routing strategy by privileging effectiveness, and combination both of them.

2) *Routing Strategy by Privileging Efficiency*

In this routing strategy, the routing indices only stores information about T-values as the goal is to locate the solutions within shortest amount of time. Given the current state of the network, the routing algorithm learns and finds an optimal routing policy from the T-values distributed over all peers in the network. Each peer *p* in the network represents its own view of the state of the network through a table that stores all the T-values, $T_p(p', r)$, where $r \in R$, a set of resource objects in the P2P systems and $p' \in N(p)$ the set of all neighbors of peer *p*. $T_p(p', r)$ is defined as the best estimated time that a query takes to reach the peer that hosts resource *r* from peer *p* through its neighbor *p'*.

The T-value, $T_p(p', r)$ can be represented by the following mathematical equation:

$$T_p(p', r) = T_{\tilde{p}'}(p'', r) + q_{\tilde{p}'} + \delta \quad (6)$$

From the equation we can derive that the minimum time needed to locate a resource r in the P2P network from the peer p through its neighbor p', is affected by 3 components:

- The time the query spends in peer p', $q_{p'}$.
- The transmission delay between p and p', δ.
- The best estimation time for $T_{p'}$ (p'', r).

The three components are used to make locally optimal routing decisions. When a peer p receives a query $q(s, r)$ looks for resource r, peer p will look its T-values table $T_p(p', r)$ to select the neighboring peer p' with the minimum $T_p(p', r)$ value. With this mechanism being used in every peer in the network, the query will be answered within the shortest amount of time.

3) *Routing Strategy by Privileging Effectiveness*

In this routing strategy, users are interested in documents that have the highest similarity to their queries. Finding such solutions may require queries to travel many peers. As a consequence, it may take a longer time to locate the solutions. The information about the estimates of similarity of documents to a particular topic can be found in R-values stored in the routing indices. As can be clearly seen in Equation 5, the R-values are updated by the highest similarity value of the documents and topic that can be achieved through neighbor peers.

As the R-values keep the best estimation of similarity, it is easy to use them for making locally optimal routing decisions. When a peer p receives a query $q(s, r)$ looks for the resource r, peer p will look its R-values table $R_p(p', r)$ to select the neighboring peer p' with the maximum $R_p(p', r)$ value. With this mechanism being used in every peer in the network, the query will produce answers displaying maximum similarity.

4) *Routing Strategy by Combining Efficiency and Effectiveness*

The general routing strategies route a request to the neighbor with the smallest T-value for a strategy seeking to optimize the response time and to the neighbor with the highest R-value (while the local R-value is smaller than the neighbors R-values) for a strategy seeking to optimize the similarity to the query. Occasionally, requests are randomly routed to allow the correction of the estimated values. Also in practice, requests are deleted from the system after they have travelled a predetermined number of hops known as the time-to-live (TTL).

A strategy seeking the combined optimization of the routing time with the similarity of retrieved documents to the query is clearly a call for trade-off. It is clear that the more exhaustive the search (therefore the longer the search), the higher the chances to locate and retrieve more similar documents. Such a strategy combines the T-values and the R-values into a single value.

Combining the R- and T-values into a single value that reflects the goodness of the peer cannot be straightforward as the two values have opposite meaning; the higher the T-value, the less efficient the system. Conversely, the higher the R-value, the more effective the system. Therefore, to get a single value that represents the goodness of a neighbor, it is required to normalize the two values into comparable values and in the same range. The T-and R-values are normalized as follows:

$$norm\_R_o(n,t) = \frac{R_o(n,t)}{\max_{y \in neighbor(o)}(R_o(y,t))} \quad (7)$$

and

$$norm\_T_o(n,t) = 1 - \frac{T_o(n,t)}{\max_{y \in neighbor(o)}(T_o(y,t))} \quad (8)$$

For every pair of T- and R-values in the routing indices, a weighted sum, $V(o, t)$, of their normalized values called the V-values or routing values are computed as follows:

$$V(o,t) = w \times norm\_T_o(n, t) + (1 - w) \times norm\_R_o(n, t) \quad (9)$$

Queries are forwarded to the neighbor with the highest routing values. A value close to 0 for w emphasizes higher similarity, while a value close to 1 emphasizes better response time.

The reader notices that the weight w needs not be a parameter fixed globally to the system nor locally to the machines but can be associated to each individual request. This allows users to indicate their preference for the combination of efficiency or response time and effectiveness or relevance.

C. *Adaptive Routing Strategies Based On Internal States*

The second approach proposes routing strategies that are adaptive to internal states. The internal states observed are domains which abstract the collection and interest of the peers. The two states are common and can be easily observed over time.

This approach uses the internal states to improve both the efficiency and effectiveness of the system by creating a social group on the basis of above internal states. Through its observable internal behavior, a peer displays traits that can be assimilated with those of an individual. If a peer appears to be specialist in some domains, this means the domains are induced by the topics of documents a peer is storing and serving. If a peer appears to be interested in some (possibly different) domains, this means the domains are induced by the topics of the queries issued by the user or users of the peer.

By exploiting the two internal states, this research aims to develop an unstructured P2P architecture in which the system adaptively learns the expertise of peers, and dynamically reorganizes itself by creating efficient communities (groups) of peers.

1) *Routing Indices Design*

In adaptive routing algorithms based on internal states, the routing indices should be able to accommodate the changes of internal behavior. The systems have to learn either the peer's expertise or the peer's interests or in more advance use, the system can observe both of them. Note that the peer's expertise and interest can be represented in the form of a set of topics.

The routing indices are implemented as a table of links. The routing index maintained by a peer *o* contains for each topic (either represents expertise or interest) values denoted by $T_o(t)$ which represents a set of links to other peers that have a high possibility of answering the queries. Peers can seamlessly join and leave the network without need of telling to the whole network. Peers joining the network can initialize the content of their routing index to the default and initial values, which is empty link. Peers leaving the network just inform their direct neighbors as they do not maintain peers that refer to them. Their absence will be eventually discovered when other peers attempt to connect with them.

Each entry in the routing indices is of the form:

$$(t, ((l_1, \alpha_{t1}), .., (l_k, \alpha_{tk})))$$

for each topic of a peer's expertise/interest, where α_{tl} is the average similarity of topic *t* with collection of peer *l*, and *k* is the number of allowed links which is parameter of the system. Figure 3 illustrates a routing index with *t* topics and *k* relevant neighbors for topic *i*.

Topics	Links
T_1	$(l_1, \alpha_{11}), .., (l_k, \alpha_{1k})$
…	….
T_t	$(l_1, \alpha_{t1}), .., (l_k, \alpha_{tk})$

Figure 3. A routing index for adaptive routing algorithms based on internal states

In addition to the neighbors in the routing index, each peer maintains a list of random neighbors. These are peers (randomly selected) with which the peer was acquainted as it joined the network or received various messages. Let us say that a peer knows a total of *N* neighbors, *S* neighbors are in the routing index and *R* = *N*-S neighbors are randomly chosen (as peers join the network and interact with other peers). *R* and *S* are parameters of the system. The peers and their neighbors form a small word graph in the sense of work done by Merugu [17].

The query is forwarded to a maximum of n neighbors ($n \leq N$). However, there are some choices to forward queries to random neighbors only, to neighbors in the routing index only, or by combining them. Thus, the query can be forwarded to *s* neighbors selected from the routing index and $r = n\text{-}s$ neighbors selected from the list of random neighbors, s and *n* are parameters of the system. If $s = 0$ the peer does not use the routing index and is similar to n-broadcast à la Gnutella (see Jovanovic [23] and Yang [6]). If $s > 0$, the peer computes the similarity between the query and the t topics in the routing index, which are weighted vectors of terms or keywords. The query is forwarded to those up-to *s* neighbors in the lists of neighbors of topics whose similarity with the query is maximum and higher than a threshold σ, which is a parameter of the system. For instance, a peer *p* has 3 topics t_1, t_2, and t_3 and their corresponding most similar peers are $\{(p_1,0.8), (p_2,0.75), (p_3,0.7)\}$, $\{(p_2,0.9), (p_4,0.83), (p_5,0.7)\}$, $\{(p_2,0.84), (p_3,0.82), (p_4,0.8)\}$ respectively in its routing index. Peer *p* receives query *q* and after computing the similarity of *q* to topics t_1, t_2, and t_3, the most similar topic to query *q* is t_1, then the system will choose top-s most similar peers corresponding t_1. Given the value of n=4 and s=2, then r = n - s = 2. Therefore the system will forward query *q* to p_1 and p_2 as the most similar peers and two random peers.

As shown in [10], [11], [12] that adaptive routing indices using reinforcement learning can efficiently and effectively route queries in unstructured networks. The following subsections explore three semantically motivated approaches to the creation and maintenance of routing indices. The three proposed algorithms differ in the way the routing indices are created and maintained. First algorithm is an adaptive routing based on the expertise of peers, i.e. the topics representative of the documents stored by a peer. Such algorithm is simply referred as Expert Groups. Second algorithm is an adaptive routing based on the interest of peers, i.e. the topics representative of the queries received by the peer (issued at the peer or received and forwarded). Such algorithm is simply referred as Interest Groups. Last algorithm is an adaptive routing by combining expertise and interest groups or we referred to as hybrid groups.

2) *Expert Groups*

In this architecture, peers are grouped according to topics representative of their content or expertise, i.e. of the documents that a peer stores, thus creating expert groups/networks. As in the architecture described in Vazirgiannis [18], peers in expert groups maintain a set of feature vectors that are centroids of the clusters of the documents, which are stored by the peers. These vectors can be obtained and maintained by online versions of vector or graph clustering algorithms (see Aslam [24] for an example of an efficient and effective online clustering algorithm). These vectors represent the abstract topics, in which the peer is an expert, i.e. it can answer queries. The vectors or topics are the entries of the routing indices. Notice that the routing index is initially empty and evolves as documents are added or removed from the peer.

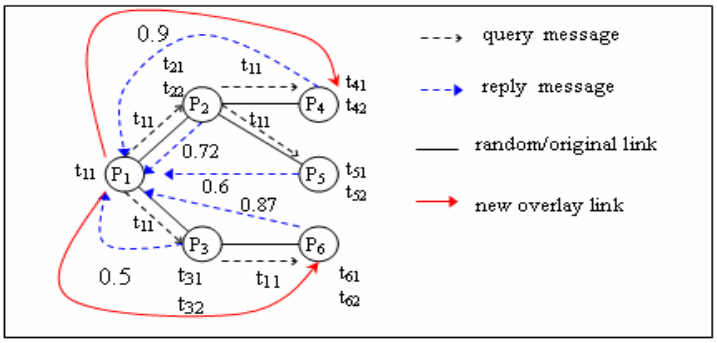

Figure 4. Illustration of links establishment in Expert Groups

When a peer joins the network, it chooses random peers to be linked. Hereafter, the peer needs to advertise its expertise to get knowledge of other peers that share similar expertise by broadcasting all its feature vectors. The depth of broadcasting is controlled by time-to-live (TTL) parameters. Figure 4 illustrates the process. Suppose the time-to-live parameter for advertising is set to 2 and P_1 that has topic t_{11} joining the network. P_1 will broadcast topic t_{11} to P_2 and P_3, and time-to-live is decremented. P_2 and P_3 will evaluate t_{11} against their own topics and send the similarity of the most similar topic to P_1, for instance 0.72 and 0.5 respectively. As the time-to-live is not yet reached, P_2 and P_3 will broadcast t_{11} to P_4, P_5 and P_6 and decrement the time-to-live accordingly. P_4, P_5 and P_6 will evaluate against their own topics and send the similarity of the most similar topic to P_1, for instance 0.9, 0.6, and 0.87 respectively. When the time-to-live is reached, the broadcasting is stopped. P_1 has received a proposal of links with similarity 0.72, 0.5, 0.9, 0.6, 0.87. If the value of parameter S is set to 2, P_1 will establish links to P_4 and P_6 as they are the top two of the most similar peers it can reach within 2 hops.

However, in order to adapt to the evolution of the network, broadcasting continues. In this way, the system learns the network status and adapts to new changes. The changes should be reflected in the routing indices by continuously exchanging peers' expertise that can be managed using several updates mechanisms: requestor, return path, forward propagation, and dual propagation. The explanation of these mechanisms can be found in the next sub section D.

When a peer receives an advertising message about another peer's expertise, it compares the message with its own by computing the similarity between the vector of expertise and the vectors of topics in the routing index. If the similarity is above a threshold ρ, which is a parameter of the system, the sending peer of the expertise is added as a neighbor corresponding to the topic in the routing index if size constraints for the routing index allow. It becomes an expert neighbor. Each topic in the routing index is associated to an explicit maximum of γ neighbors. If the peer already has γ neighbors for the topic concerned, it replaces one of expert neighbors with the smallest similarity to the topic with the new candidate neighbor, provided that the similarity of the new candidate neighbor is higher than the one to be replaced; otherwise, it ignores the new candidate.

If the similarity is above the threshold, conversely the receiving peer advertises its topics to the sending peer, which, in turn, considers it for inclusion in the list of expert neighbors in the routing index. Since the routing index is dynamically modified, the topology of the network of peers and their groups that it defines continuously evolves. Peers are dynamically grouped according to their expertise.

3) Interest Groups

In the second adaptive routing algorithm based on internal states, this research designs an architecture in which peers are grouped according to topics representative of their interest, i.e. of the queries that are issued or forwarded by the peer, thus creating interest groups. In this architecture, which is called as interest groups, the evolution of the network is different than in the previous scenario, as the interest of peers, i.e. the queries that it issues or forwards, can only be observed over time.

The routing indices contain direct links to other peers, which have answers for particular topic. It is initially empty and evolves as the queries elapsed/received/forwarded to other peers. Figure 5 shows the mechanism. Upon receiving a query, a peer will search its own database first. If it can satisfy the query, it will send the result to the requestor otherwise it will broadcast the query to the peers that have the answer (if they exist in the routing index) or broadcast to random nodes.

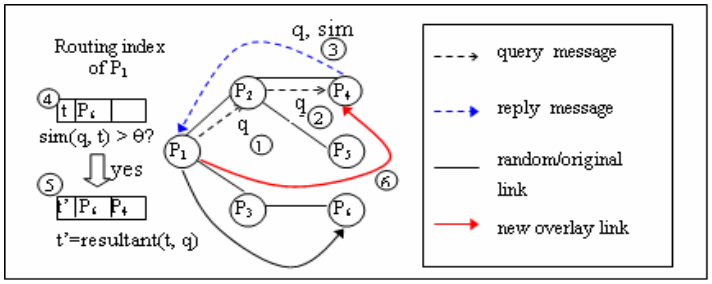

Figure 5. Illustration of links establishment in Interest Groups

When a peer receives an answer from another peer, it compares the respective query for the answer with its own by computing the similarity of the vector of query with the vector of topics in the routing index. If the similarity is above a threshold ρ, which is a parameter of the system, the answering peer of the query is added as a neighbor corresponding to the topic in the routing index (if size constraints for the routing index allow). As the vector of query and the corresponding vector of topic in the routing index are considered similar, the vector of topic in the routing index can be updated by taking the resultant between both of them. Such an approach reduces the number of vectors that should be stored in the routing index thus maintaining the scalability of the system.

Each topic in the index is associated to an explicit maximum of γ neighbors. If the peer already has γ neighbors for the topic concerned, it replaces one of neighbors with the smallest similarity to the topic with the new candidate neighbor, provided that the similarity of the candidate neighbor to the topic is higher than that about to be replaced; otherwise, it ignores the new candidate neighbor. The same mechanisms in section 3.4, which are requestor, return path, forward and forward propagation and dual propagation can be applied to links exchange upon sending/receiving queries. By these mechanisms, the system will evolve and will dynamically and adaptively change its topology and follow the behavior of users' interest. Thus peers are dynamically grouped according to users' interest.

4) Hybrid Groups

Finally, the last approach provides a design of hybrid architecture combining neighbors obtained from both the expert groups and the interest groups. The system is designed by adding more entries in the routing indices of expert groups. In the standard expert groups, the routing indices only contain vectors representing the expertise of the peers. In the hybrid groups, routing indices of expert groups are added with interest links. This design has an anthropomorphic value: individuals seeking knowledge will navigate networks of acquaintances characterized by their expertise and interest.

D. Propagation Strategies of Routing Indices

The routing indices are designed to store information about internal and external states in local point of view of the peers, as it is quite expensive for collecting the whole states of the network. Basically peers can exchange their local view of the internal and external states in order to get estimation of the whole states of the network. Thus global view of the network states is emergent from interaction among peers. In this way, peers learn the network states so that they can take appropriate actions to respond to the users' queries.

The proposed adaptive routing algorithms adopt the concept of propagation to disseminate and learn the changes of internal and external states. The changes then are reflected in the routing indices based on the received information. This research studies four mechanisms for propagating changes in internal and external states, which are requestor, return path, forward propagation, and dual propagation. The propagations are conducted during communication among peers in the network. The exchanged information can be in forms of following:

- The estimation of routing time in case of adaptive routing strategy by privileging efficiency
- The estimation of similarity in case of adaptive routing strategy by privileging effectiveness
- The expertise links of corresponding queries in case of expert groups
- The interest links of corresponding queries in case of interest groups

Q-Routing algorithm uses the forward propagation, whereas the dual Q-Routing uses dual propagation. Figure 6 shows the illustrations of the four studied mechanisms. Below are the explanations of each strategy based on the figures:

- In the requestor strategy, a peer answering a query propagates the corresponding entry in the routing index to the peer that issued the query. Suppose P_1 issues a query q, and P_x has an answer for query q and sends to P_1. Upon sending the answer, P_x will also bring information from its routing index to P_1.
- In the return path strategy, a peer answering a query propagates the corresponding entry of the query in the routing index to all peers on the return path to the peer that issued the query. Suppose the query q from P_1 has been forwarded through P_2, and P_4 has the solution. P_4 will bring information from its routing index to P_1 and P_2.
- In the forward propagation strategy, a peer receiving a query (but not necessarily answering it) propagates the corresponding entry of the query in the routing index to the peer that sent the query. Suppose the query q from P_1 is forwarded through P_2, P_2 will answer the query q and bring information from the routing index to P_1. If P_2 cannot satisfy the query, it will forward the query to its neighbor, for instance to P_4. P_4 will answer the query as it has the solutions and bring information of its routing index to P_2. The same process occurs until the query meets its time-to-live or the solutions are found.
- In the dual propagation strategy, in addition to a backward propagation, the peer also propagates the corresponding entry of the query in the routing index to the peers it contacts (a backward propagation). Upon peer P sending the query to neighbor, it also brings information of its routing index and asks the neighbor to update its routing index (if applicable). The neighbor replies and brings information from its routing index to peer P.

Liu et al [25] proposed strategies of propagation by employing requestor and return path methods. Forward and dual propagation are exploited in routing algorithms proposed by Kumar et al [26], [27].

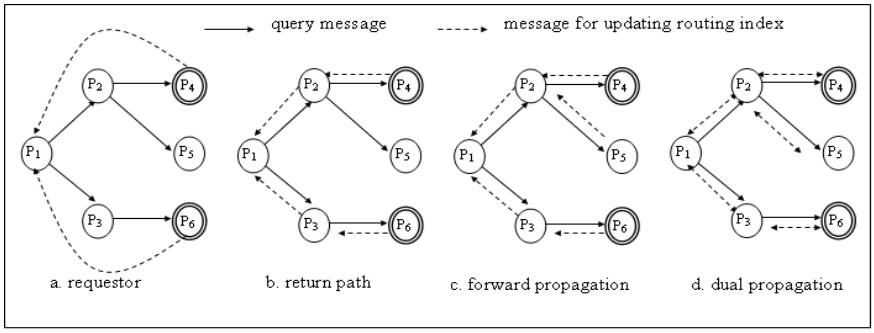

Figure 6. Illustration of the four strategies of changes propagation

E. *Pruning of Routing Indices*

The proposed adaptive algorithms also rely on routing indices to select the best peer(s) that will forward the query. The number of entries in the routing indices depends linearly on the number of topics in the whole system, in the case of adaptive routing strategy based on external states, interest groups and hybrid group. Storing all possible topics in the routing indices creates a scalability problem due to their potentially big numbers which make them unmanageable. An approach using routing indices is only scalable if the routing indices are of manageable size.

In this sense, the routing indices can be viewed as caches. As the caches may have limited size comparing to number of information that have to be stored, it is required that the system only stores important information

accessed frequently. Fortunately, there are several page replacement algorithms that can be used to control information moved in and out of the cache. For this purpose of replacement strategies, this research also studies the popularity of resources in the local context.

The following subsections explain the proposed algorithms to prune the size of routing indices. The pruning strategies adopt standard replacement strategies which are Random, Least Recently Used (LRU), and Least Frequently Used (LFU) method. Later, more advanced strategies can be used that perhaps will lead better performance. Below are the explanations of the used strategies:

- Random Method

  This strategy can be viewed as the simplest strategy to manage routing indices. Given the fixed size of the routing index, the strategy will randomly remove one of the entries in the routing index to be replaced by a new entry (topic). This strategy is very simple and does not require any additional information.

- Least Recently Used (LRU) Method

  This method is based on the Least Recently Used (LRU) algorithm. Each peer learns the local popularity of resources by examining queries it receives. Each routing index only stores *n* topics which are most recently requested and their corresponding information. When a peer receives a request on topic *t*, its routing index is updated by replacing a topic that has been least recently requested with *t* and its corresponding information.

- Least Frequently Used (LFU) Method

  This method is based on the Least Frequently Used (LFU) algorithm. The removal of any entry is based on the popularity of the topics. Each peer learns the local popularity (instead of global popularity) of topics by examining queries it receives. Each routing index only stores *n* topics which are most frequently requested. When a peer receives a request for topic *t*, its routing index is updated with the replacement of a topic that has been least frequently requested with *t* and its corresponding information. A column is added to the routing index to store frequency of access to the topics. The value in the column is set to 1 when a topic is put into the routing index for the first time or when it replaces the least frequently used topic. The value is incremented if the topic is in the routing index already.

IV. EXPERIMENTAL RESULTS AND PERFORMANCE ANALYSES

In the following section, we present our experiment results. The simulations use the PeerSim simulator [28] as a platform of the simulations. Some researchers also conducted their simulations on top of the Peersim simulator, among others are OverStat [29], [30], SG-1 [31] and T-Man [32]. Therefore, we believe it is appropriate to choose Peersim as the simulator for the experiments in this research.

A. Simulation Parameters

The simulations use a WireKOut graph structure as the topology of the network. A WireKOut is a graph with *n* vertices, each of which is connected to its nearest *k* neighbors, which are chosen randomly. The experiments use 4.000 vertices (peer) and a *k* value of 4. This relatively small number is meant to reflect the small world effect. The delays between peers are assigned a value between 1 and 100; these values are generated randomly.

We use symbols to represent keywords in the simulations. Hereafter, the terminology of symbol is referred to as keyword. Topics and documents are represented as a set of symbols. These simulations use 1,000 symbols to represent 1,000 keywords. These keywords are used to create 500 root topics as a base to create topics and queries for peers in the network. A topic consists of 5 weighted keywords. The weight of each keyword is assigned a randomly selected value between 0 and 1 inclusively. The function used to compute similarity between topics is Euclidian-distance. Each peer is assigned with 1 to 3 topics by varying the weight of keywords in the root topics. The weight of each keyword is varied over the weight in the root topics. Each peer is also assigned with 1 to 10 documents for each topic in the peer. The documents are created by varying the peer's topics with the same fashion as creation of topics in peers from root topics.

Queries are generated using Zipfian distribution with parameter θ to reflect the popularity of each topic in the network. The default value of θ in the simulation is 0.5. Setting of parameter θ to 1 will lead to uniform distribution. The smaller the value of θ, the fewer the number of popular topics will be, i.e., few topics are accessed frequently and many topics are accessed infrequently. The values of θ are varied in some simulation scenarios.

Query originators are determined using [X,Y] distribution. The distribution means that X% of queries must be similar to peers' topics and Y% of queries are determined randomly. A distribution of 0/100 reflects truly random distributions of query originators. The default value of the distribution in our simulation is [50,50]. In some

simulation scenarios, the values of X and Y are varied to simulate the behavior of users' queries. The higher the value of X, the higher the users tend to request documents which are similar to their collection.

A query is assigned a time-to-live (TTL) value. These simulations use two different values of TTL. In the first simulation for experimenting adaptive routing based on external state, the default value of TTL is set to 500. In the second experiment of adaptive routing algorithms based on internal states, the default value for TTL is set to 5. This means a query can travel a maximum of 500 hops from its origin for the first scenarios and 5 hops from its origin for the second scenarios. In the first experiments, the TTL is set to high value as the nature of the first proposed algorithms that only forward the query to single neighbor. As the solutions possibly located far from the requestor, setting the TTL to a small value will cause many queries failing to satisfy the query. In contrast, in the second experiments, the TTL is set to small value as the nature of the algorithms that use broadcast mechanism to forward queries. By setting the TTL to 5 and each peer has 4 neighbors, a query will traverse to around 1,364 peers ($4^1 + 4^2 + 4^3 + 4^4 + 4^5$) or around 1/3 of the size of the network to locate the solutions. Some scenarios vary values of TTL to study the impact of TTL to the performance of the proposed routing algorithms.

The queries are also assigned a MaxWaitTime to reflect the maximum time a query waits for results from other peers. Answers that come to the requestor after MaxWaitTime will be ignored from the evaluation. Table 1 summarizes the parameters of the simulations.

TABLE I. PARAMETERS OF SIMULATIONS

Parameter	Value
Network topology	WireKOut
Number of peers	4,000
Number of random neighbors per peer	4
Number of expert/interest neighbors per peer	4
Number of keywords (symbols)	1,000
Number of root topics	500
Number of keywords per root topic	5
Weight of each keyword	[0..1] (Assigned randomly)
Number of topics per peer	Max. 3 keywords (Derived from root topics by varying the weight of each keyword)
Number of documents per topic n each peer	Max. 10 (Derived from the corresponding topic by varying the weight of each keyword)
Default Zipfian θ	0.5 (Controlling the popularity of topics)
Default [X,Y] distribution	[50,50] (Controlling the relevance of queries to the peers' expertise)
Number of queries per simulation time	25
Simulation time	5,000
Default queries' TTL	5 (Default value for adaptive routing strategies based on internal states)
	500 (Default value for adaptive routing strategies based on external states)
Maximum time for peers to wait answers from other peers	150 (Default value for adaptive routing strategies based on internal states)
	3,000 (Default value for adaptive routing strategies based on external states)

B. Experimental Results

To measure the performance of the proposed algorithms, some measurements have been defined as follows:

- Number of messages is defined as the accumulative number of messages generated for running the simulation. The messages counted may be in the form of messages for routing queries, messages for updating routing index and messages for finding similar peers.
- Number of answered queries is defined as the number of queries answered by at least one peer.
- Average response time of top-10 documents is defined as the average response time for locating answers in the top-10 documents. As in the previous measure, if no peer answers the query, the response time will be set to MaxWaitTime.
- Average similarity of top-10 documents is defined as the average similarity of the top-10 documents with the highest similarity to the query. If no peer answers the query, the average similarity will be set to zero.

In our first experiments, we investigate the performance of adaptive routing strategies based on external states. As we mentioned in our research design, our research discussed some issues including: routing index design, replacement strategy, and effect of routing index size to the routing performance. These issues will be explored detailer in this first experiment. In the second experiment, we investigate the performance of adaptive routing strategies based on internal states. In this second experiment, we focus in studying the performance of the proposed strategies.

1) *Experiment Results on Adaptive Routing Strategies Based on External States*

The experiments are conducted by setting a parameter w to various values representing users' preference. The simulation uses dual propagation strategies as it shows the best result among other strategy. The values of w are set to 0, 0.2, 0.4, 0.6, 0.8 and 1. Setting value of w to 0 reflects the users who look for effectiveness. Setting value of w to 1 reflects the users who look for efficiency. Setting value of w between 0 and 1 will take effect of compromising between efficiency and effectiveness.

Figures 7 to 10 show the results of the experiments. As seen in the figures, when value of w is set to 1, the solution can be found in the shortest amount of time regardless the similarity of retrieved documents to the query, i.e. emphasizing to the efficiency. When value of w is set to 0, the solution can be found with the highest similarity to the query regardless the time needed to locate the solution, i.e. emphasizing to the effectiveness. Setting different values of the w indeed produce the expected gradual effect on routing time and similarity of answers to the query: routing time converges faster at smaller values as w varies from 0 to 1; average similarity varies from 0.5 to 0.9 as w varies from 1 to 0. Expected trends also happen on the number of generated messages and the number of queries answered and satisfied. As the values of w vary from 0 to 1, the system generates fewer messages and answers and satisfies more queries.

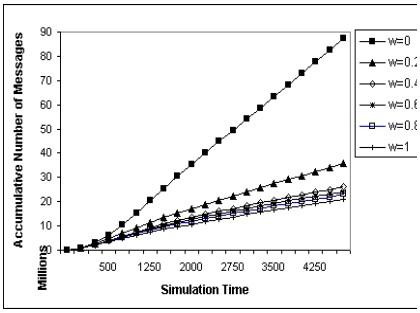

Figure 7. Number of messages generated for various value of w

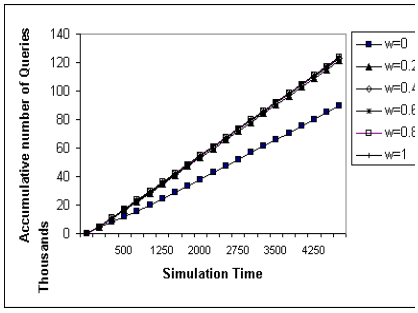

Figure 8. Number of answered queries for various value of w

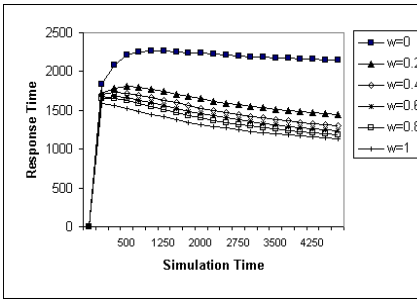

Figure 9. Average response time of the top-10 documents for various value of w

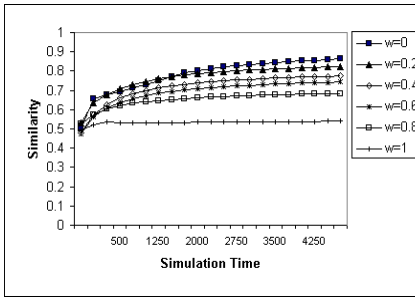

Figure 10. Average similarity of the top-10 documents for various value of w

In real situations, not all topics have equal popularity. The distribution model widely used to represent such a situation is the Zipfian distribution. The distribution is controlled by the θ parameter. The values of θ are controlled by these values: 0, 0.2, 0.4, 0.6, 0.8, and 1. The smaller the value of θ, the more skewed the distribution is (i.e. fewer popular topics). This experiment studies the effect of the number of popular topics to the performance of the system, particularly when the sizes of the routing indices are limited. For this purpose, the routing indices are pruned up to 50% of their original size. The parameter w is set to 1.

The experiment results show that popularity that is more skewed improves the convergence of the response time as can be seen in Figure 11 and 12. The smaller value of Zipfian θ also reduces the number of generated messages as can be seen in Figure 11. The smaller the value of Zipfian θ, the fewer the popular topics is. As the number of popular topics is small, the system has more chance to load all those popular topics in the routing indices. The information in those routing indices is also frequently updated thus reflecting the latest states of the

network. Hence, the performance of the system in terms of efficiency is improved over time as can be seen in Figure 12. This experiment shows when many users tend to search for particular popular topics, the system performs better in achieving stability. Thus, the system offers better performance in satisfying such popular queries.

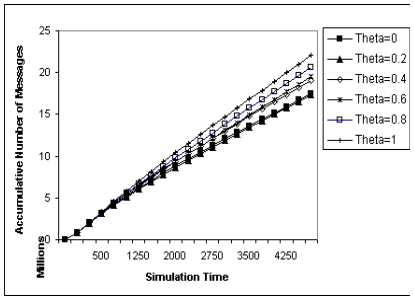

Figure 11. Number of messages generated for various Zipfian θ

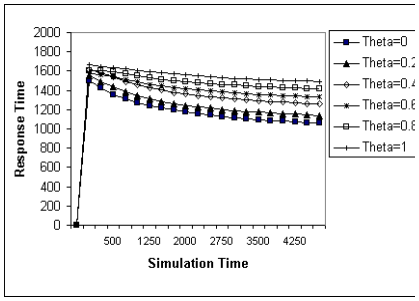

Figure 12. Average response time of the top-10 documents for various Zipfian θ

We also investigate the performance of the propagation strategies for routing indices. The propagation of routing indices entries are the key of having adaptive routing algorithms. It is expected by appropriate propagation strategies, the system learns faster in order to improve the performance the routing. Original Q-Routing uses only forward propagation to update the routing indices. Some improvements can be made to speed up stabilizing the routing indices. These experiments study the performance of the proposed propagation strategies, which are requestor, return path, forward propagation and dual propagation strategy.

Figures 13 and 14 show the experiment results using these different propagation strategies. In terms of routing time, dual propagation outperforms other strategies, whereas the requestor strategy is the worst one as it learns the slowest. As can be seen in the Figure 14, the routing time for the requestor strategy increases then it is stable at a certain routing time. The trend of routing time for other strategies is different as they are initially increasing, but later they are capable to decrease. Among others, dual propagation strategy learns the fastest thus offering the best routing time to locate the answers.

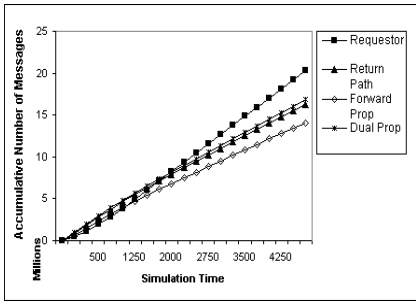

Figure 13. Number of messages generated for various propagation strategies

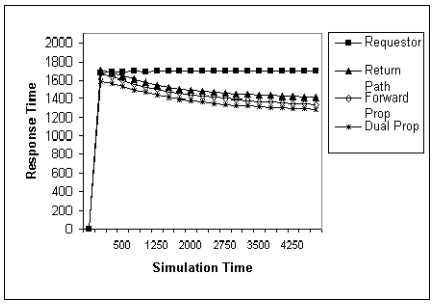

Figure 14. Average response time of the top-10 documents for various propagation strategies

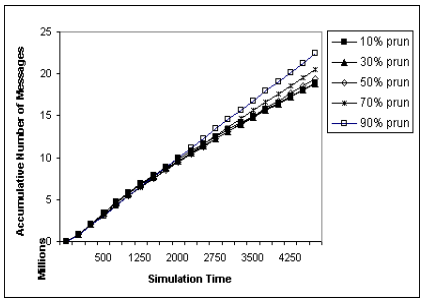

Figure 15. Number of messages generated for various pruning levels

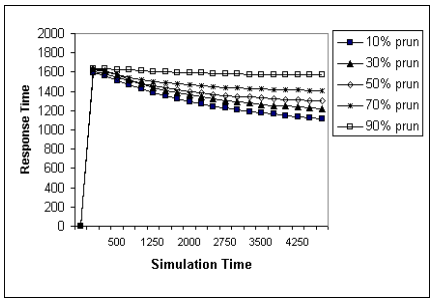

Figure 16. Average response time of the top-10 documents for various pruning levels

In terms of number of generated messages as can be seen in Figure 13, the requestor strategy generates the highest number of messages even though it sends propagation messages to the requestor only. This is possibly caused by the system that learns very slowly, which is as a consequence, the system needs farther hops to locate the answers. Forward propagation strategy offers the lowest number of generated messages. The forward propagation strategy outperforms the dual propagation strategy as the dual propagation strategy needs to double its messages for routing indices propagation.

The last experiments study the effect of pruning size to the performance of the proposed routing algorithms. The pruning levels of the routing indices are set to 10%, 30%, 50%, 70%, and 90%. The pruning level of X% is determined by pruning the size of the routing indices up to X% of the root topic size. For example, as these simulations use 500 root topics, by pruning level of 10% it means the system only keeps 450 entries in the routing index of each peer. Figures 15 and 16 show the results of the experiments.

The figures show the higher the pruning factor, the longer the system to find answers from the network. Increasing pruning factor also generates more messages, as the system needs longer paths to locate the answers, particularly for those non popular topics that may not be kept in the routing indices of peers. In terms of number of answered queries, the system is capable to answer almost an equal number of queries for all pruning levels, given the parameter of the experiments. With a smaller value of queries' TTL, it is expected that the smaller the size of the routing indices, the smaller the number of answered queries is, as many queries travel to the network without guidance from the routing indices.

These experiments show that keeping more information in the routing indices offers better performance. More queries are forwarded to the correct paths as the routing indices offer greater possibility to provide the information corresponds to the topics of the queries. Thus, the most ideal situation is to have each peer with an unlimited size of the routing index; it offers the best performance.

2) *Experiment Results on Adaptive Routing Strategies Based on Internal States*

First experiment evaluates the performance of expert networks by varying number of random and expert links for broadcasting. Figures 17 to 19 show the performance of the system for various values of R, the number of random neighbors, to which a query is forwarded, and S, the number of expert neighbors in the routing index to which a query is forwarded. The performance baseline is random broadcast strategy ala Gnutella (S=0, R=4).

Figure 17 shows the number of messages decreases with more neighbors from the routing indices as searches are more focused to peers with same expertise. Figure 18 shows the response time to locate the top-10 documents.. As seen in the figure, the more the users use expert neighbors, the shorter the time needed to obtain the first answer. Figure 19 shows the average similarity of the top-10 documents. The Figure shows that broadcasting queries to all expert neighbors improves the similarity of documents found. Overall, using more expert neighbors improves the performance of the routing.

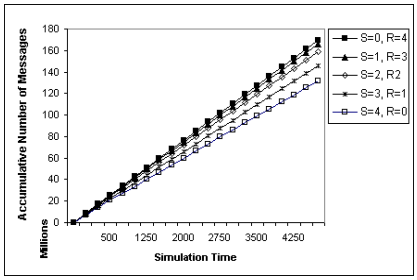

Figure 17. Number of messages generated for varying number of random links

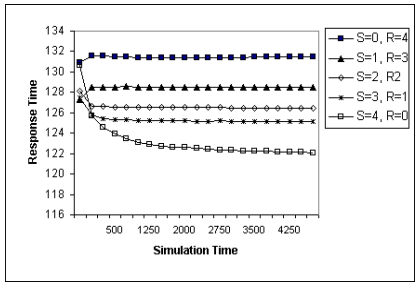

Figure 18. Average response time of the top-10 documents for varying number of random links

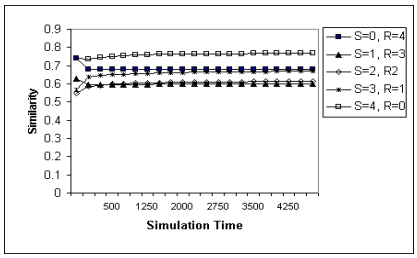

Figure 19. Average similarity of the top-10 documents for varying number of random links

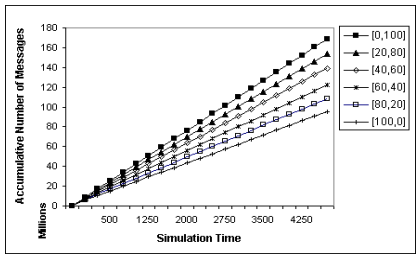

Figure 20. Number of messages generated for various [X,Y]

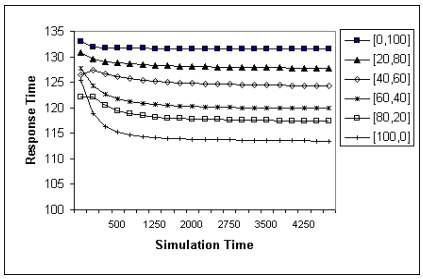

Figure 21. Average response time of the top-10 documents for various [X,Y]

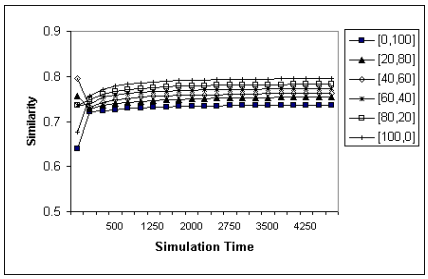

Figure 22. Average similarity of the top-10 documents for various [X,Y]

Second experiments on expert groups are conducted by varying the proportion [X,Y] of related queries to unrelated queries to the peers expertise. The baseline is random originator, i.e. proportion [0,100], where the queries' topics are assigned randomly.

Figures 20 to 22 show that more related queries to the peers' expertise improve the performance of the system in all performance measurements. As seen in Figure 20, more related queries to the peers' expertise can reduce the number of generated messages. Figure 21 presents the response time of the system. Indeed, when more related queries elapse in the system, the performance is better. However, the similarity of the top-10 documents is slightly improved. As shown in the Figure 22, from the proportion of [0,100] to [100,0], the average similarity of the top-10 documents only increases in order less than 0.1.

We also experimented on interest groups. As explained in the research design, in interest groups each peer will examine queries elapsed or forwarded to it. Thus, it probably requires a big size of the routing index as in the adaptive routing strategies based on external states. Therefore this routing technique is expected to have similar properties with the adaptive routing strategies based on external states particularly in accordance with routing index management.

First, we compared the performance of interest groups by varying value of the queries' TTLs. Figures 23 to 26 show the results of the experiments. Again, as expected, setting longer queries' TTLs produces a better performance at the cost of generating more messages.

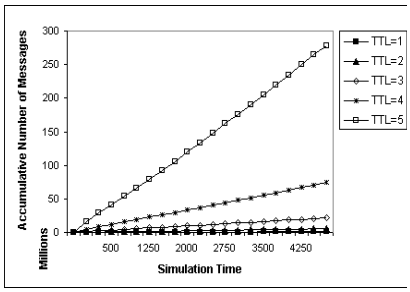

Figure 23. Number of messages generated for various queries' TTLs

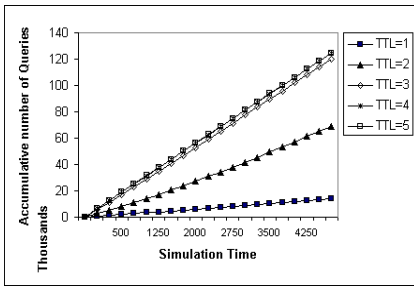

Figure 24. Number of answered queries for various queries' TTLs

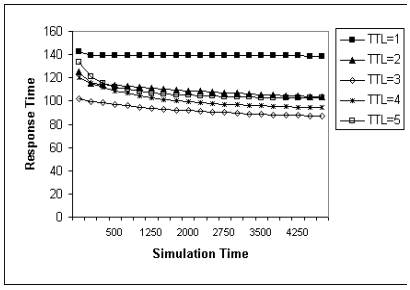

Figure 25. Average response time of top-10 documents for various queries' TTLs

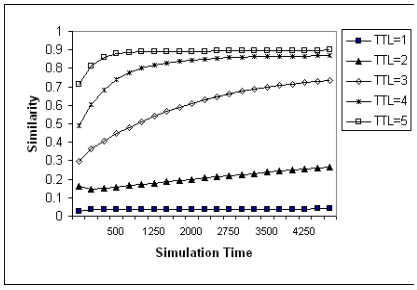

Figure 26. Average similarity of top-10 documents for various queries' TTLs

The experiment results show that the users cannot use the queries' TTLs that are too low as the system will not learn, but the users also do not need to set the queries' TTLs to a very high value as this offers decreasing improvement on performance. At a particular value of queries' TTLs, the system reaches its maximum performance. Afterward, increasing the queries' TTLs generates more messages with lesser improvement on the system performance.

Last, we showed the simulation results of the three proposed strategies in this category compared to other similar algorithms. There are two algorithms which are used as baselines of comparison: peer clustering and firework query model by Cheuk Hang et al [19] and interest-based locality by Kunwadee Sripanidkulchai et al [33]. Both algorithms are similar to the proposed adaptive algorithms. However, both of them do not have the capability of managing the routing indices. The network topology in [19] is also considered fixed as the reorganization of network topology is only conducted when the peers join the system. In [33], the reorganization of network topology is performed only when peers receive answers from other peers. The approach is quite similar with the proposed adaptive algorithm when using requestor propagation strategy.

As can be seen in Figure 27, the proposed algorithm by Hang produces the smallest number of messages, whereas the expert groups and algorithm by Sripanidkulchai generate almost an equal number of messages. As previously explained, interest groups produces the highest number of messages. The algorithm by Hang very efficient in generating messages as its mechanism that only forwards queries to a single neighbor when reaching the cluster hosts the answer of the queries. In the proposed grouping mechanisms and algorithm by Sripanidkulchai, the system broadcasts the queries to all neighbors that make them producing messages higher than the algorithm by Hang.

In term of number of answered queries, the algorithm by Hang only answers about a half of queries elapsed by the peers as shown in Figure 28. In algorithm by Hang, once a peer joins the network and gets a shortcut for the most similar peer, the network topology is not changed. The algorithm by Hang requires the peers to advertise their expertise to the whole system to be efficient and effective as they indeed rely only on this advertisement phase to group peers.

Figure 29 compares the performance of the algorithms in terms of routing time to locate top-10 similar documents to the queries. The proposed interest and hybrid groups outperform other strategies. The interest and hybrid groups also perform better in retrieving documents with high similarities to the queries than other strategies as can be seen in Figure 30. Again, algorithm by Hang performs the worst and only capable to retrieve documents with similarity around 0.35.

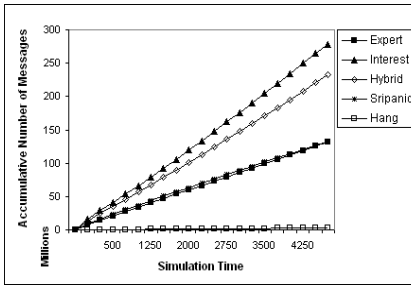

Figure 27. Number of messages generated for all compared algorithms

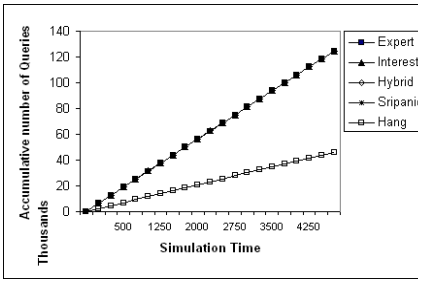

Figure 28. Number of answered queries for all compared algorithms

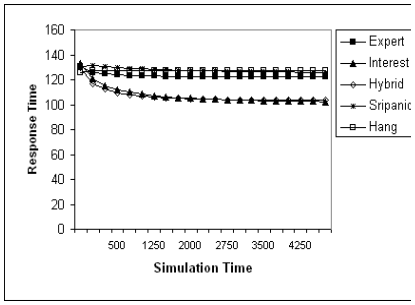

Figure 29. Average response time of top-10 documents for all compared algorithms

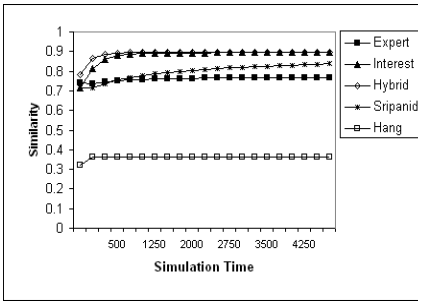

Figure 30. Average similarity of top-10 documents for all compared algorithm

From the explanation, it can be concluded that the proposed adaptive routing strategies are competitive enough. They are capable to locate answers in efficient time and effective retrieval. In implementing this adaptive routing, the users should be aware the possibility of a high number of messages as the impact of propagation of expertise and interest. However, it is expected by applying heuristic constraint as in adaptive routing based on external states (instead of broadcasting), the problem of a high number of message can be solved.

When comparing results of the two strategies, it also can be seen that the adaptive routing strategies based on internal states show their outstanding performance over the adaptive routing strategies based on external states. In the adaptive routing strategies based on internal states, the system reorganizes its original topology by allowing peers to create additional links (shortcuts) to other peers that share similar expertise/interest. Thus it forms groups as a metaphor of social behavior. This mechanism enables peer jumping to long-distance peers that have the answers. Without shortcuts, when searching for peers hosted the solution, peers should travel to other peers in the network hop-by-hop until they find the peers hosted the answers. With shortcuts, peers can search the solution directly to their shortcuts, as shortcuts point to other peers that share similar expertise/interest to these peers. Thus it reduces number of intermediate peers that should be contacted to locate solutions. It explains why the proposed adaptive routing algorithms based on internal states perform better than the proposed adaptive routing algorithms based on external states.

V. CONCLUSIONS AND FUTURE WORKS

All experimental results show how the proposed algorithms can learn and adapt to internal and external changes. In the beginning of the simulation, the performances in terms of routing time and similarity are unstable. But over time, the systems improve the performance and reach stability. Each proposed strategy has its own advantages and disadvantages. However, the simulation results show that the proposed adaptive routing strategies based on internal states outperform the proposed adaptive routing strategy based on external states in terms of routing time and similarity. In the adaptive routing based on external states, the user needs thousands of simulation time to get the first answer of the solutions. In the adaptive routing based on internal states, the users only need tens of simulation time to get the first answer of solutions.

Comparison with other algorithms also shows that the proposed adaptive routing strategies based on internal states are competitive enough. The proposed algorithms are proven capable of providing better performance than the proposed algorithms by Hang et al [19] and Sripanidkulchai et al [33]. The main issue in the proposed approaches that needs to be considered is the high number of messages generated. Combining routing algorithm between adaptive routing based on external and internal states is a good candidate for reducing the messages. Another customization of algorithm also can be used for retrieving documents in any forms such as text documents, image documents and so forth.

This research has proposed adaptive routing strategies that can be implemented for efficient and effective retrieval in P2P systems. The proposed algorithms are implemented in Peersim simulator. Measuring the exact effectiveness and efficiency of the proposed algorithms needs real environment and real documents. There are also some opportunities to improve the efficiency of the proposed algorithms. Here the future works that are considered important to be done to get more comprehensive system:

- Conduct experiments using real documents. These experiments are important to measure precisely the effectiveness of the proposed algorithms. Some collections such as TREC have million of documents with relevant judgments for provided queries.
- Implement the algorithms in real application. Some possible applications can be developed using the proposed strategies are file-sharing system, social behavior in e-Learning system, and so forth. In the context of e-Learning system, we can develop mechanism of personalization by examining the discussion contents, collections and so fort, thus creating social network of students and teachers.
- Combine the proposed strategies (adaptive routing strategies based on external and internal states) into single systems. The benefit of the first strategies using adaptation of Q-Routing algorithm can be incorporated into the proposed grouping strategies. It is expected that the combination strategy can reduce traffic of the network thus improving efficiency.
- Improve the routing strategies by incorporating other aspects. A possible approach that can be explored is incorporating the replication strategy to the system. Through elegant way of replication, it is expected that the system will have more chance to locate the solution thus improving the routing time and similarity of retrieved documents.

REFERENCES

[1] NAPSTER, http://www.napster.com

[2] GNUTELLA, http://gnutella.wego.com

[3] FastTrack, http://www.fasttrack.nu

[4] Clarke, I., Sandberg, O., Wiley, B., Hong, T.W.: Freenet: A Distributed Anonymous Information Storage and Retrieval System. In: Lecture Notes in Computer Science, (2001)

[5] Wikipedia, http://en.wikipedia.org/wiki/Peer-to-peer

[6] Yang, B., Garcia-Molina, H.: Efficient Search in Peer-to-Peer Network. In: 22nd International Conference on Distributed Computing Systems, (2002)

[7] Stoica, I., Morris, R., Karger, D., Kaashoek, F., Balakrishnan, H.: Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. In: International Conference of ACM SIGCOMM, pp. 149–160, (2001)

[8] Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A Scalable Content Addressable Network. In: International Conference of ACM SIGCOMM, (2001)

[9] Druschel, P., Rowstron, A.: Past: A Large-Scale, Persistent Peer-To-Peer Storage Utility. In: 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII), (2001)

[10] Liau, C.Y., Hidayanto, A.N., Bressan, S.: Adaptive Peer-To-Peer Routing With Proximity. In: 14th International Conference on Database and Expert Systems Applications, (2003)

[11] Hidayanto, A.N., Bressan, S., Liau, C.Y., Hasibuan, Z.A.: Adaptive Double Routing Indices: Combining Effectiveness and Efficiency in P2P Systems. In: 15th International Conference on Database and Expert Systems Applications, (2004)

[12] Hidayanto, A.N., Bressan, S., Hasibuan, Z.A.: Exploiting Local Popularity to Prune Routing Indices in P2P Systems. In: 15th International Workshop on Database and Expert Systems Applications, (2005)

[13] Hidayanto, A.N., Santoso, H.B., Aji, R.F., Bressan, S.: Community Access Point in Indonesia: Improving Access to Quality Information and Promoting Local Potentials. In: 5th International Conference of E-Business, (2006)

[14] Hidayanto, A.N., Bressan, S.: Towards a Society of Peers: Expert and Interest Groups in Peer-to-Peer System. In: On The Move (OTM) Workshop, Portugal, (2007)

[15] Aberer, K., Hauswirth, M.: An Overview on Peer-to-Peer Information Systems. In: WDAS, Carleton Scientific, (2002)

[16] Crespo, A., Garcia-Molina, H.: Routing Indices for Peer-To-Peer Systems. In: International Conference on Distributed Computing Systems, July, (2002)

[17] Merugu, S., Srinivasan, S., Zegura, E.: Adding Structure to Unstructured Peer-To-Peer Networks: the Use of Small-World Graphs. J. Parallel and Distributed Computing, Vol. 65, No. 2, pp. 142-153, February, (2005)

[18] Vazirgiannis, M., Nørv°ag, K., Doulkeridis, C.: Peer-to-Peer Clustering for Semantic Overlay Network Generation. In: 6th International Workshop on Pattern Recognition in Information Systems, Cyprus, May, (2006)

[19] Hang Ng, C., Sia, K.C.: Peer Clustering and Firework Query Model. In: 11th International World Wide Web Conference, Hawaii, (2002)

[20] Ramaswamy, L., Gedik, B., Liu, L.: Connectivity Based Node Clustering in Decentralized Peer-to-Peer Networks. In: 3rd International Conference of Peer-to-Peer Computing, (2003)

[21] Kalogeraki, V., Gunopulos, D., Yazti, D.Z.: A Local Search Mechanism for Peer-to-Peer Network. In: 11th International Conference on Information and Knowledge Management, Washington, (2002)

[22] Littman, M., Boyan, J.: A Distributed Reinforcement Learning Scheme for Network Routing. In: International Workshop on Applications of Neural Networks to Telecommunications, (1993)

[23] Jovanovic, M., Annexstein, F., Berman, K.: Scalability Issues in Large Peer-To-Peer Networks -- A Case Study of Gnutella. Technical Report, University of Cincinnati, (2001)

[24] Aslam, J., Pelekhov, K., Rus, D.: The Star Clustering Algorithm. J. Graph Algorithms and Applications, 8(1) 95–129, (2004)

[25] Liu, X., Liu, Y., Xiao, L.: Improving Query Response Delivery Quality in Peer-to-Peer Systems. J. IEEE Transactions on Parallel and Distributed Systems, Vol. 17. No. 11, pp. 1000-9999, November, (2006)

[26] Kumar, S., Mikkulainen, R.: Dual reinforcement q-routing: An on-line adaptive routing algorithm. In: Artificial Neural Networks in Engineering Conference, (1998)

[27] Kumar, S., Miikkulainen, R.: Confidence Based Dual Reinforcement Q-routing: An Adaptive Online Network Routing Algorithm. In 16th International Joint Conference on Artificial Intelligence, (1999)

[28] PeerSim Simulator, http://peersim.sourceforge.net/

[29] Jelasity, M., Montresor, A.: Epidemic-Style Proactive Aggregation in Large Overlay Networks. In: 24th International Conference on Distributed Computing Systems, Japan, (2004)

[30] Montresor, A., Jelasity, M., Babaoglu, O.: Robust Aggregation Protocols for Large-Scale Overlay Networks. In: International Conference on Dependable Systems and Networks, Italy, (2004)

[31] Montresor, A.: A Robust Protocol for Building Superpeer Overlay Topologies. In: 4th International Conference on Peer-to-Peer Computing, Switzerland, August, (2004)

[32] Jelasity, M., Babaoglu, O.: T-Man: Gossip-Based Overlay Topology Management. In: Engineering Self-Organizing Applications, (2005)

[33] Sripanidkulchai, K., Maggs, B.M., Zhang, H.: Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems. In: INFOCOM, (2003)