Discovering Joining Nodes and Detecting Leaving Nodes in the iTrust Membership Protocol

Yung-Ting Chuang, P. M. Melliar-Smith, L. E. Moser, Isaí Michel Lombera

***Abstract*—We present a membership protocol for the iTrust search and retrieval network. In iTrust, a source node distributes metadata together with the URL for a document to a subset of randomly chosen nodes in its local view of the membership. A requesting node distributes a request (query) containing keywords to a subset of randomly chosen nodes in its local view of the membership. If a node receives a request such that the keywords in the request match metadata that it holds, then it sends to the requesting node a response containing the URL, so that the requesting node can retrieve the document from the source node.**

The membership protocol for iTrust allows each member to have its own local view of the membership. A requesting node detects a non-operational node by not receiving a response from that node within a timeout, or by receiving an error code from TCP. Likewise, a requesting node discovers a newly joined node when it receives a response from a node that has included that newly joined node in its view. Our performance evaluation demonstrates that, for appropriate values of the parameters, the iTrust membership protocol discovers joining nodes and detects leaving nodes to maintain a local view of the membership that is close to the actual membership.

***Index Terms*—membership protocol, membership churn, decentralized search and retrieval, trustworthy information access, iTrust.**

I. Introduction

UNBIASED and uncensored retrieval of information over the Internet is crucial for modern society. Currently, Internet search and retrieval exploits centralized search engines for reasons of efficiency and economy of scale. Unfortunately, it is easy to cause a centralized search engine to conceal or censor information. To ensure the free flow of information over the Internet, an alternative to centralized search – an effective decentralized, distributed search – must be provided.

The iTrust system [1], [4], [10], [11] is a decentralized and distributed search and retrieval system, that is designed to make it difficult to censor or filter information accessed over the Internet. Metadata and requests are "public," because nodes must be able to match the keywords in the requests against the metadata that they hold. The communication cost for iTrust is greater than that for centralized search engines. However, if users suspect that information important to them is being censored or suppressed, they should be willing to incur that extra cost. Nonetheless, in iTrust, we try to minimize any additional unnecessary cost.

In this paper, we describe a membership protocol for the iTrust search and retrieval network. The *membership* consists of the nodes that participate in the iTrust network (also referred to as the *participating nodes*). Each node has a *local view* of the *membership*, which approximates the actual membership of the iTrust network.

An extensive literature on membership exists, but most of that work is not relevant to iTrust. The membership protocol for iTrust is simpler and less costly than prior membership protocols [3], because iTrust does not attempt to achieve an agreed accurate membership based on a consensus algorithm, which is known to be impossible [2].

II. The iTrust Messaging Protocol

First, we briefly describe the iTrust messaging protocol, because the iTrust membership protocol is dependent on it. The steps involved in the iTrust messaging protocol are given below and are illustrated in Figure 1.

1) A node with information to share (a *source node*) distributes metadata for that information to a subset of nodes chosen uniformly at random from its local view of the membership.
2) A node that needs information (a *requesting node*) distributes a request (query) containing keywords for that information to a subset of nodes chosen uniformly at random from its local view of the membership.
3) When a node receives a request containing keywords that match metadata that it holds, an *encounter* occurs and the matching node provides, to the requesting node, the URL for the information, so that the requesting node can retrieve the information from the source node.
4) The requesting node then retrieves the information from the source node using the URL provided by the matching node.

A *match* between the keywords in a request received by a node and the metadata held by that node might be an exact match or a partial match, or might correspond to synonyms.

Distribution of the metadata and the requests to relatively few nodes suffices to achieve a high probability of a match. As we have shown in [10], in an iTrust membership with N nodes, distribution of the metadata to $M = 2\sqrt{N}$ nodes and distribution of the requests to $R = 2\sqrt{N}$ nodes results in a probability of a match that exceeds $1 - e^{-4} \approx 0.9817$.

Manuscript received December 6, 2012; revised January 9, 2013.

This research was supported in part by the U.S. National Science Foundation under grant number NSF CNS 10-16193.

Y. T. Chuang is with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA, 93106, e-mail: ytchuang@ece.ucsb.edu

P. M. Melliar-Smith, L. E. Moser, and I. Michel Lombera are with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA, 93106, email: pmms@ece.ucsb.edu, moser@ece.ucsb.edu, imichel@ece.ucsb.edu

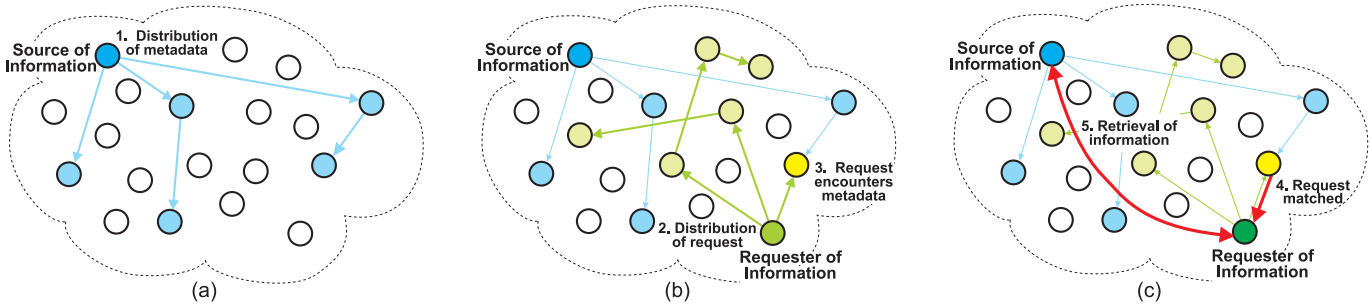

Fig. 1. (a) A source node distributes metadata, describing its information, to a subset of nodes chosen uniformly at random from its local view of the membership. (b) A requesting node distributes its request (query) to a subset of nodes chosen uniformly at random from its local view of the membership. (c) One of the nodes matches the metadata and the request, and reports the match to the requesting node, which then retrieves the information from the source node.

III. The iTrust Membership Protocol

In an iTrust network, nodes can *join* the membership at any time; likewise, nodes can *leave* the membership at any time, either voluntarily or by crashing.

The membership protocol for iTrust does not aim to achieve an agreed accurate membership that is the same for all members. Rather, it allows each member to have its own local view of the membership, but aims to keep that local view close to the actual membership.

A. Joining the Membership

To join the membership, a node must first obtain the address of a bootstrapping node. To obtain the address of the bootstrapping node, the node uses mechanisms outside the iTrust network, such as conventional Web search, E-mail, Twitter, printed publications, *etc*.

The steps involved when a node joins the membership are given below, and are illustrated in Figure 2.

1) The joining node contacts the boostrapping node, using the address it previously obtained, to obtain the bootstrapping node's current view of the membership.
2) The joining node then publishes its joining the membership to a subset of nodes chosen uniformly at random from the current view of the membership it obtained from the bootstrapping node.
3) Those nodes then add the new node to their local views of the membership.

Another node learns about the new node when it receives a response from a node that is aware of the new node.

B. Leaving the Membership

A node may leave the membership either voluntarily, or because it is faulty. The steps involved in leaving the membership are simple:

1) To leave the membership, a node just leaves, without publishing its leaving.

Over time, each node individually discovers the departure of nodes when it sends requests to nodes that do not respond. It is not appropriate to allow a node to publish the departure of another node, because doing so can enable a malicious node to cause a requesting node to remove many nodes from its local view of the membership.

C. Updating the Membership

In the messaging protocol described in Section II, a requesting node expects to receive response messages from the matching nodes. Other nodes that don't have a match are not required to send a response to the requesting node. We now modify that messaging protocol to enable a requesting node to discover newly joined nodes and to detect non-operational (leaving) nodes from the responses to its requests.

Now, the requesting node expects each node to which it sent a request to respond to its request with its recently joined member(s), regardless of whether or not it has a match. A matching node sends in its response to the requesting node not only the URL of the document at the source node but also its recently joined member(s). If it does not have a match, a node simply responds to the requesting node with its recently joined member(s). Thus, the requesting node discovers not only the URLs of the documents, but also some newly joined nodes through the responses to its requests.

If the requesting node doesn't receive a response from a node within a timeout period or it receives an error code from TCP, then the non-responding node is considered to have left the membership or to be faulty and the requesting node removes that node from its view of the membership.

The steps involved in updating a requesting node's view of the membership are given below, and are illustrated in Figure 2(b) and (c).

1) A requesting node distributes its request to a subset of randomly chosen nodes in its local view of the membership.
2) A node that receives the request compares the keywords in the request with the metadata that it holds. If it finds a match, the matching node responds to the requesting node with a message that contains the URL of the associated information and also its recently joined member(s). A node that doesn't find a match responds to the requesting node with a message that contains its recently joined member(s).
3) When the requesting node receives the responses, it adds the newly joined members obtained from the other nodes to its view of the membership.
4) If the requesting node does not receive a response from a node to which it sent a request, before a timeout occurs, or if it receives an error code from TCP, then the non-responding node is considered to have left the membership or to be faulty, and the requesting

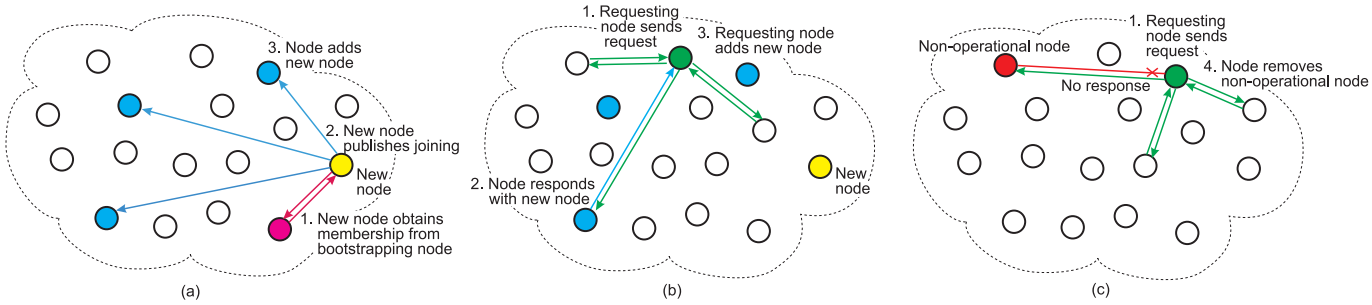

Fig. 2. (a) A node joins the membership by first contacting a bootstrapping node to obtain that node's current view of the membership, and then publishing its joining to randomly chosen nodes in its current view of the membership. (b) A requesting node distributes a request to nodes randomly chosen from its current view of the membership. A node that receives the request returns the members(s) that it recently added to its membership. A matching node also returns the URL of the document at the source node to the requesting node. The requesting node adds the new nodes to its view of the membership. (c) The requesting node does not receive a response to its request from a non-operational node. The requesting node sees a timeout expire or gets an error code from TCP, and then removes the node from its membership.

node removes that node from its local view of the membership.

If the requesting node is also a source node then, after receiving the responses to its request, it distributes its metadata to additional nodes according to the following steps:

1) The requesting (source) node calculates the number of nodes to which it needs to distribute its metadata, according to its current view of the membership.
2) Next, the requesting node subtracts the number of nodes to which it previously distributed metadata from the number it just calculated in the previous step.
3) The requesting node then distributes its metadata to that many additional nodes, chosen at random from its current view of the membership, but to which it had not sent the metadata previously.

For example, suppose that a requesting node currently has $N = 1024$ nodes in its current view of the membership. It distributes its request to $2\sqrt{1024} = 64$ nodes, chosen at random. Suppose that only 58 nodes reply to the requesting node. From the responses to this request, the requesting node detects that there are $64 - 58 = 6$ non-operational nodes. Suppose further that, as a result of receiving the responses from the 58 nodes, the requesting node adds 40 new nodes to its membership. Consequently, the requesting node now has $N = 1024 - 6 + 40 = 1058$ nodes in its membership. If the requesting node is also a source node, then it distributes its metadata to $2\sqrt{1058} - 2\sqrt{1024} = 65 - 64 = 1$ more node in its local view of the membership.

IV. Performance Evaluation

A. Simulation Experiments

To evaluate the iTrust membership protocol, we performed experiments using a simulation of iTrust. The simulation allows us to evaluate the performance of the membership protocol, whereas a real-world deployment of iTrust would not allow us to do so, because in the simulation we can control the joining rate and the leaving rate of the nodes. Moreover, we can compare a node's current view of the membership against the actual membership.

Before we start the simulation program, we initialize the value of N. Next, the program adds all of the nodes to each node's membership so that each node has the complete initial membership. Then, for each time step, nodes make requests while other nodes are joining and leaving the membership. Finally, the program compares each node's view of the membership against the actual membership.

B. Membership Churn

Membership churn refers to nodes joining and leaving the membership, and is represented by the following rates:

- JR: The Joining Rate, the number of nodes that join the membership per time unit. For example, $JR = 50$ means that 50 nodes join the membership per time unit.
- LR: The Leaving Rate, the number of nodes that leave the membership per time unit. For example, $LR = 50$ means that 50 nodes leave the membership per time unit.

When there is a lot of membership churn, both JR and LR are high. When the membership is stable, both JR and LR are low. These rates are an important consideration for the membership protocol.

C. Parameters of the iTrust Membership Protocol

The parameters of the iTrust membership protocol are the following:

- RR: The Requesting Rate, the number of times a node sends a request message to $2\sqrt{N}$ nodes per time unit. For example, $RR = 10$ means that a node sends 10 distinct request message per time unit, each of which is sent to $2\sqrt{N}$ nodes.
- $LastJ$: The Last Joined members, the number of recently joined members a node may report back to the requesting node. For example, $LastJ = 2$ tells a node to report its two most recently joined members to the requesting node.

D. Performance Metrics

For the performance evaluation, we use the following performance metrics:

- LND: The Leaving Not Detected nodes, the proportion of non-operational nodes that a requesting node has not detected in a given time unit.
- JND: The Joining Not Discovered nodes, the proportion of newly joined nodes that a requesting node has not discovered in a given time unit.

E. Tuning the Parameters

1) Determining an Appropriate Value of $LastJ$: The parameter $LastJ$ is used to discover newly joined nodes, but it doesn't help much in detecting leaving or non-operational nodes, as discussed below.

In our membership protocol, a requesting node distributes its request to $2\sqrt{N}$ nodes chosen at random from its local view of the membership. Initially, we required those nodes to return their entire memberships to the requesting node, and the requesting node to update its membership accordingly. The problem is that the requesting node obtains some non-operational nodes from other nodes that haven't yet discovered that those nodes are non-operational. Thus, the requesting node adds back into its membership too many non-operational nodes, including nodes that it recently removed. The requesting node can't distinguish between such non-operational nodes that left the membership, and nodes that left the membership and recently re-joined.

There are several possible solutions to this problem. One solution is that, once a requesting node has obtained the memberships from the other nodes, it sends a "verify" message to confirm whether or not each of those nodes is operational. Such a solution consumes a lot of communication bandwidth. Another solution is to require the $2\sqrt{N}$ nodes to return their most recently joined members, rather than their entire memberships, to the requesting node. We adopt the latter solution and let $LastJ$ be the maximum number of recently joined members that a node may return. We investigate how $LastJ$ affects LND (Leaves Not Detected) and JND (Joins Not Discovered).

We consider a scenario in which there are $N = 1024$ nodes with a high leaving rate ($LR = 500$), a high joining rate ($JR = 500$), and a low requesting rate ($RR = 10$). Figure 3 shows the graphs for LND and JND over time for this scenario with $LastJ = 1$, $LastJ = 2$ and $LastJ = 3$. Increasing $LastJ$ from $LastJ = 1$ in the left graph to $LastJ = 2$ in the middle graph, we see that JND decreases but that LND increases. Increasing $LastJ$ further from $LastJ = 2$ in the middle graph to $LastJ = 3$ in the right graph, we see that JND decreases further and that LND increases further to about 0.9.

Thus, increasing $LastJ$ definitely helps the requesting node to discover more joining nodes as it issues more requests. However, increasing $LastJ$ also causes the requesting node to add back into its local view of the membership too many non-operational nodes. The reason is that a requesting node can detect at most $2\sqrt{N}$ non-operational nodes from the responses to each of its requests. However, if $LastJ = 2$, then the requesting node can discover up to $2{*}2\sqrt{N} = 4\sqrt{N}$ newly joined nodes from the responses to each of its requests. Doing so for $LastJ = 2$ actually increases LND with worse results than for $LastJ = 1$. Similarly, when $LastJ = 3$, LND increases even more compared to when $LastJ = 2$. We conclude that $LastJ = 1$ is an appropriate value for our further experiments. Thus, a node can discover about the same number of non-operational and newly joined nodes from the responses to each of its requests, and doesn't add back into its local view of the membership too many non-operational nodes.

2) Determining an Appropriate Value of RR: As we have just seen, increasing $LastJ$ does not help to detect non-operational or leaving nodes. Thus, we now investigate how increasing the requesting rate RR affects the detection of non-operational nodes.

Figure 4 shows graphs for LND and JND over time for $RR = 10$, $RR = 50$ and $RR = 100$, where $N = 1024$ initially, $LastJ = 1$, $JR = 500$ and $LR = 500$. When RR is increased from $RR = 10$ in the left graph to $RR = 50$ in the middle graph, both JND and LND greatly decrease. In fact, JND decreases to almost zero in the $RR = 50$ case. When RR is increased further from $RR = 50$ in the middle graph to $RR = 100$ in the right graph, LND decreases and becomes close to 0.1, and JND still remains close to 0.

From Figure 3 and Figure 4, we conclude that increasing RR is more effective in decreasing both LND and JND than increasing $LastJ$. Thus, we use $RR = 100$ in our further experiments.

3) Investigating Different Values of LR and JR: Here, we use the values $LastJ = 1$ and $RR = 100$ as determined by our previous experiments, where $N = 1024$ initially.

Figure 5 shows graphs for LND and JND over time for $LR = JR = 500$, $LR = JR = 1000$ and $LR = JR = 2000$. In the left graph of Figure 5, we see that JND remains zero from the beginning to the end of the scenario and that LND slowly increases to 0.08 and remains there until the end of the scenario.

When LR and JR are increased from $LR = JR = 500$ in the left graph to $LR = JR = 1000$ in the middle graph, JND still remains close to zero from the beginning to the end of the scenario, but LND increases to about 0.113. We had expected that LND would also increase, but it does not increase very much, even though LR and JR are twice as large as they were in the left graph.

When LR and JR are further increased from $LR = JR = 1000$ in the middle graph to $LR = JR = 2000$ in the right graph, JND increases a little to about 0.02 and LND now increases to about 0.22, perhaps a little higher than we would wish. Even though LR and JR are much higher than in the left graph and the middle graph, both JND and LND still remain reasonably small.

In addition, we investigated LND and JND for $LR = JR = 2000$ and $RR = 200$ and also for $LR = JR = 3000$ and $RR = 300$. In these two cases, we obtain similar results to those in the middle graph of Figure 4 and in the middle graph of Figure 5. Note that, in all of these cases, $RR = LR/10 = JR/10$. Currently, we are investigating a membership protocol for iTrust that modifies RR adaptively, so that JND and LND remain small, while minimizing the additional overhead due to larger values of RR.

From Figure 3, Figure 4 and Figure 5, we conclude that, for appropriate choices of LR, JR and RR, our membership protocol is effective, in that a node maintains its local view of the membership very close to the actual membership, even when the membership churn is high.

V. Related Work

Peer-to-peer networks for distributed search have been characterized as structured or unstructured [12]. The structured approach requires the nodes to be organized in an overlay network, based on a distributed hash table, tree, ring, etc. The unstructured approach uses randomization and/or gossiping to distribute the metadata (or data) and the

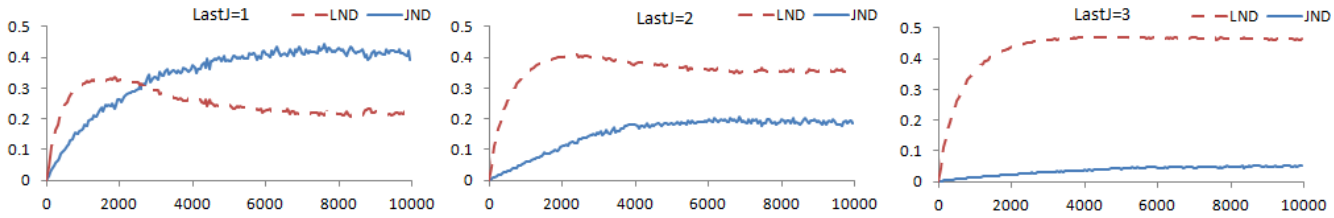

Fig. 3. Graphs showing LND and JND over time for $LastJ = 1$, $LastJ = 2$ and $LastJ = 3$, where $N = 1024$ initially, $RR = 10$, $LR = 500$ and $JR = 500$

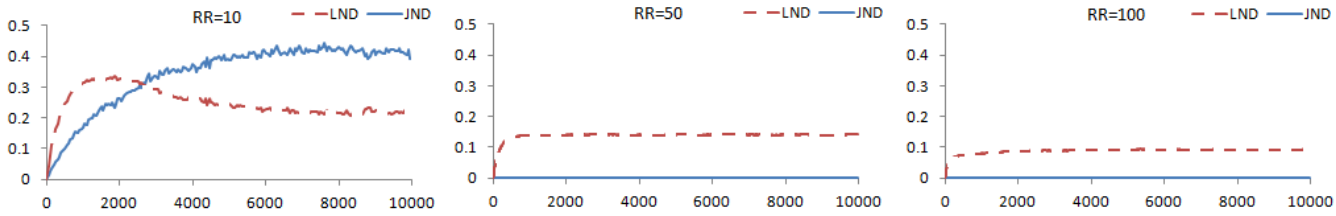

Fig. 4. Graphs showing LND and JND over time for $RR = 10$, $RR = 50$ and $RR = 100$, where $N = 1024$ initially, $LastJ = 1$, $LR = 500$ and $JR = 500$.

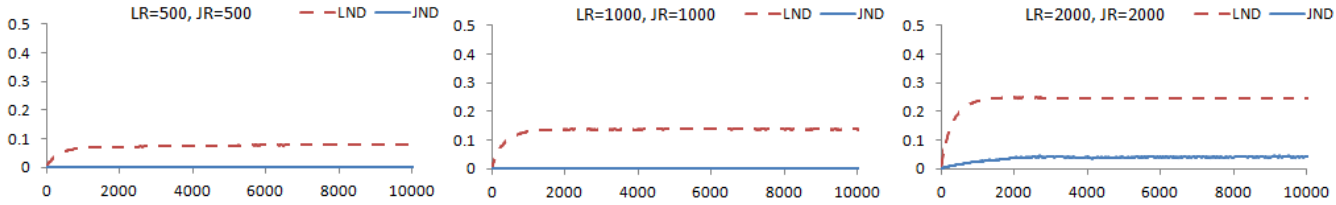

Fig. 5. Graphs showing LND and JND over time for $JR = LR = 500$, $JR = LR = 1000$, and $JR = LR = 2000$, where $N = 1024$ initially, $LastJ = 1$ and $RR = 100$.

queries to nodes in the network. The iTrust system uses the unstructured approach.

Cohen and Shenker [5] demonstrate that square root replication is theoretically optimal for minimizing search traffic, and replicate objects based on the access frequencies (popularities) of the objects. Lv *et al.* [9], in conjunction with Cohen and Shenker, use square root replication, and adaptively adjust the replication degree based on the query rate. iTrust likewise exploits square root replication, but distributes the metadata and requests uniformly at random, so that popular nodes are not more vulnerable to attacks.

Prior work on membership has focused on an agreed accurate membership in the presence of unreliable processors and unreliable communication. Chandra *et al.* [2] show that it is impossible to achieve an agreed accurate membership. Chockler *et al.* [3] provide a comprehensive survey of membership protocols and group communication systems, and of their formal specifications. Schiper and Toueg [13] provide an elegant formalization of the membership problem that distinguishes between the problem of maintaining and agreeing on a set of members and the problem of determining which processes are working and should be members. Our membership protocol for iTrust is simpler and less costly than such prior work. It does not aim to achieve an agreed accurate membership based on a consensus algorithm. Rather, it allows each member to have its own local view of the membership, but aims to keep that local view close to the actual membership.

In BubbleStorm [7], [14], when a node wishes to join the network, it finds an existing connection between two peers and interposes the joining node between them. When a node leaves the network, it re-connects those two peers before leaving. Thus, BubbleStorm preserves a fixed node degree at all of the nodes. Our membership protocol for iTrust does not restrict the nodes to a fixed node degree but, rather, allows each node to maintain its local view of the membership.

SCAMP [6] is a peer-to-peer membership service for gossip-based protocols that operates in a decentralized and self-configured manner, where no peer has global knowledge of the membership. A node that wishes to join the membership notifies some nodes in the network to add it to their memberships. Similarly, a node that wishes to leave the membership notifies some nodes to remove it from their memberships. To prevent a node from becoming isolated, each node periodically tries to discover new nodes when it does not receive messages in a given time period. In contrast, our membership protocol for iTrust places more emphasis on maintaining the membership when the churn rate is high.

Zage *et al.* [16] present a network-aware and distributed membership protocol that improves the overall performance of a peer-to-peer overlay network by biasing neighbor selections towards beneficial nodes, based on multiple system metrics and network social patterns. In the iTrust membership protocol, nodes do not maintain their views of the membership through biased neighbor selections. Rather, they discover newly joined and leaving nodes by communication

with randomly chosen nodes in the normal course of search and retrieval

Liu *et al.* [8] describe a novel age-based membership protocol with a conservative neighbor maintenance scheme under churn, to retain desirable properties such as a low network diameter and clustering coefficient. Thus, a bootstrapping node recommends to a newly joining node only the nodes that have remained in its membership for a long time. However, with this strategy, a newly joining node might not discover the other nodes in the membership very quickly. In the iTrust membership protocol, a boostrapping node sends its entire membership to a newly joining node.

Voulgaris *et al.* [15] present an inexpensive membership management protocol for unstructured peer-to-peer overlay networks. The goal of CYCLON is to have each node maintain a small and fixed-size neighbor list. The authors describe a shuffling protocol for large networks and provide an experimental analysis in which they examine the clustering coefficient and node degree distribution. The iTrust membership protocol differs from CYCLON in that each node tries to discover as many nodes as possible to include in its local view of the membership. In [1], we have investigated the use of neighborhoods and de-clustering (shuffling) for very large iTrust networks.

VI. Conclusion

We have presented a membership protocol for the iTrust search and retrieval network. The membership protocol for iTrust allows each member to maintain its own local view of the membership, but aims to keep that local view close to the actual membership. A node that receives a request sends to the requesting node a response that contains newly joined node(s) in its local view of the membership. The node also sends in its response the URL of the document, if the keywords in the query match metadata that it holds.

A requesting node discovers newly joined nodes from the responses it receives to its requests, which include newly joined node(s) in the responding node's view of the membership. Likewise, a requesting node detects nodes that are non-operational or that have left the membership, as a result of not having received responses from those nodes before a timeout occurs, or on receiving an error code from TCP. Thus, the iTrust membership protocol exploits messages already required by the iTrust messaging protocol for search and retrieval, rather than additional messages used only for membership. As our performance evaluation demonstrates, for appropriate values of the parameters, the iTrust membership protocol discovers joining nodes and detects leaving nodes to maintain a local view of the membership that is close to the actual membership.

In the future, we plan to continue our investigation of the performance of the iTrust membership protocol in other scenarios. We also plan to develop an adaptive membership protocol for iTrust, where each node increases its rate of sending requests as it detects non-operational nodes that do not respond to requests and as it discovers new nodes that recently joined the membership. In addition, we plan to address nodes that exhibit malicious behavior when responding to requests.

References

[1] C. M. Badger, L. E. Moser, P. M. Melliar-Smith, I. Michel Lombera, and Y. T. Chuang. Declustering the iTrust search and retrieval network to increase trustworthiness. In *Proceedings of the 8th International Conference on Web Information Systems and Technologies*, pages 312–322, Porto, Portugal, April 2012.

[2] T. D. Chandra, V. Hadzilacos, S. Toueg, and B. Charron-Bost. On the impossibility of group membership. In *Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing*, pages 322–330, 1996.

[3] G. V. Chockler, I. Keidar, and R. Vitenberg. Group communication specifications: A comprehensive study. *ACM Computing Surveys*, 33(4):427–469, 2001.

[4] Y. T. Chuang, I. Michel Lombera, L. E. Moser, and P. M. Melliar-Smith. Trustworthy distributed search and retrieval over the Internet. In *Proceedings of the International Conference on Internet Computing*, pages 169–175, Las Vegas, NV, July 2011.

[5] E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks. In *Proceedings of the ACM Special Interest Group on Data Communications Conference*, pages 177–190, Pittsburgh, PA, August 2002.

[6] A. Ganesh, A. M. Kermarrec, and L. Massoulie. Peer-to-peer membership management for gossip-based protocols. *IEEE Transactions on Computers*, 52(2):139–149, February 2003.

[7] C. Leng, W. W. Terpstra, B. Kemme, W. Stannat, and A. P. Buchmann. Maintaining replicas in unstructured P2P systems. In *Proceedings of the ACM Conference on Emerging Networking Experiments and Technologies*, page 19, Madrid, Spain, December 2008.

[8] H. Liu, X. Liu, W. Song, and W. Wen. An age-based membership protocol against strong churn in unstructured P2P networks. In *Proceedings of the 2011 International Conference on Network Computing and Information Security*, volume 2, pages 195–200, Guilin, China, May 2011.

[9] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th International Conference on Supercomputing*, pages 84–95, Baltimore, MD, June 2002.

[10] P. M. Melliar-Smith, L. E. Moser, I. Michel Lombera, and Y. T. Chuang. iTrust: Trustworthy information publication, search and retrieval. In *Proceedings of the 13th International Conference on Distributed Computing and Networking*, pages 351–366, Hong Kong, China, January 2012.

[11] I. Michel Lombera, Y. T. Chuang, P. M. Melliar-Smith, and L. E. Moser. Trustworthy distribution and retrieval of information over HTTP and the Internet. In *Proceedings of the 3rd International Conference on the Evolving Internet*, pages 7–13, Luxembourg City, Luxembourg, June 2011.

[12] J. Mischke and B. Stiller. A methodology for the design of distributed search in P2P middleware. *IEEE Network*, 18(1):30–37, 2004.

[13] A. Schiper and S. Toueg. From set membership to group membership: A separation of concerns. *IEEE Transactions on Dependable and Secure Computing*, 3(1):2–12, 2006.

[14] W. W. Terpstra, J. Kangasharju, C. Leng, and A. P. Buchmann. Bubblestorm: Resilient, probabilistic, and exhaustive peer-to-peer search. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures and Protocols for Computer Communications*, pages 49–60, Kyoto, Japan, August 2007.

[15] S. Voulgaris, D. Gavidia, and M. Van Steen. CYCLON: Inexpensive membership management for unstructured P2P overlays. *Journal of Network and Systems Management*, 13(2):197–217, June 2005.

[16] D. Zage, C. Livadas, and E. M. Schooler. A network-aware distributed membership protocol for collaborative defense. In *Proceedings of the International Conference on Computational Science and Engineering*, volume 4, pages 1123–1130, 2009.