Trustworthy Distributed Search and Retrieval over the Internet

Yung-Ting Chuang, Isaí Michel Lombera, L. E. Moser, P. M. Melliar-Smith
Department of Electrical and Computer Engineering
University of California, Santa Barbara
Santa Barbara, CA 93106 USA

Abstract— *This paper describes iTrust, a novel distributed search and retrieval system that provides trustworthy access to information over the Internet. Nodes with information to distribute transmit their metadata to nodes that are selected at random from a set of participating nodes. Similarly, nodes seeking information distribute their requests to nodes that are selected at random from the set of participating nodes. When a node receives a request, the node tries to match the metadata in the request with the metadata that it holds. If the node has a match, it supplies a URL for the information to the requesting node, which then retrieves the information from the source node. The paper describes our implementation of iTrust, and provides a performance evaluation of iTrust, based on both analysis and simulation using our implementation. Distribution of metadata and requests to relatively few nodes suffices to achieve a high probability of a match.*

Keywords: trustworthy distributed Internet search retrieval

1. Introduction

Our modern world relies heavily on the ability to publish, search for, and retrieve information over the Internet, which has created a highly distributed information society, distributed in both the sources of information and the uses of information. For reasons of efficiency and scalability, conventional search and retrieval over the Internet employs centralized search engines.

Unfortunately, centralized Internet search engines can be tampered with easily by their administrators to bias the results, concealing or censoring information. The experience of history, and even of today, indicates that we cannot rely on centralized Internet search to remain unbiased forever. Perhaps, the moment at which we are most dependent on our ability to communicate over the Internet is also the moment at which centralized Internet search is most likely to be compromised. It is important to ensure that a trustworthy distributed search and retrieval system for the Internet is available when it is needed, even though a user normally uses a conventional centralized search engine.

The iTrust system, described in this paper, is a novel distributed search and retrieval system that provides access to information over the Internet. The iTrust system involves distribution of metadata and requests, matching of requests and metadata, and retrieval of information corresponding to metadata. iTrust has no centralized mechanisms that can be tampered with easily by a small group of administrators. iTrust is inevitably more costly in bandwidth, processing and storage than a centralized search engine. Individuals who are concerned about a risk of censorship ought to find that cost acceptable.

The iTrust system is deployed on a set of participating nodes in the Internet (also referred to as the membership). iTrust distributes both metadata that describes information, and requests for information, to a random subset of the participating nodes in the Internet. Because the metadata and the requests are distributed to nodes that are chosen at random from among all of the participating nodes, no one node or small group of nodes can suppress or censor information.

In the iTrust system, source nodes produce information and publish that information to make it available to other participating nodes. The source nodes create metadata keywords for their information, and communicate that metadata, together with a URL, to a subset of the participating nodes that are chosen at random, as shown in Figure 1.

Requesting (querying) nodes generate requests (queries), containing metadata keywords for information that they seek to retrieve. The requesting nodes distribute their requests to a subset of the participating nodes that are chosen at random, as shown in Figure 2.

If a participating node receives a request, it compares the metadata in the request with the metadata that it holds. If the metadata match, which we call an encounter or a match, the matching node returns to the requesting node the URL that the source node included with the metadata, as shown in Figure 3. The requesting node then uses the URL to retrieve the information from the source node.

The random distribution of the metadata and the requests achieves a high probability of a match, even when the metadata and the requests are distributed to relatively few nodes. Moreover, the probability of a match remains high even when some of the participating nodes (even some of the randomly chosen nodes) are subverted or non-operational.

The rest of this paper is organized as follows. Section 2 describes the implementation of the iTrust system. Performance evaluation results, based on both analysis and simulation using the iTrust implementation, are presented in Section 3. Section 4 presents related work, and Section 5 presents conclusions and future work.

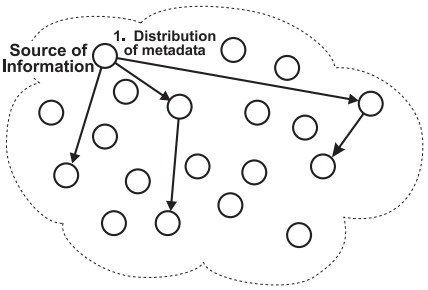

Fig. 1: A source node distributes metadata, describing its information, to randomly selected nodes in the membership.

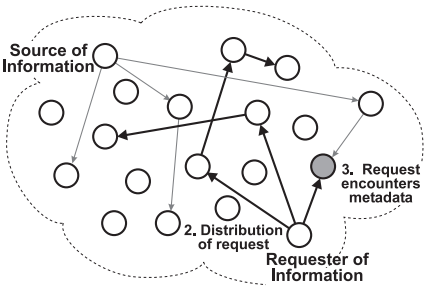

Fig. 2: A requesting node distributes its request to randomly selected nodes in the membership. One of the nodes has both the metadata and the request and, thus, an encounter occurs.

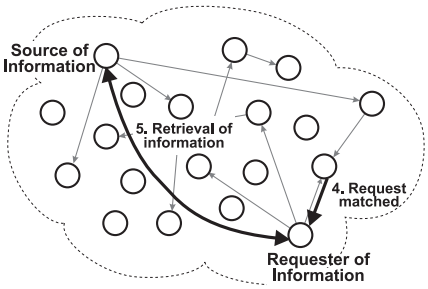

Fig. 3: A participating node matches the metadata and the request and reports the match to the requester, which then retrieves the information from the source node.

2. The iTrust System

The iTrust system on a node consists of three distinct components that interact with each other to distribute metadata and requests and to retrieve information (resources). Figure 4 shows the three components: the Web server foundation, the application infrastructure, and the public interface. Arrows on connecting lines indicate the direction of information flow. The following subsections describe these three components and their interactions.

2.1 Web Server Foundation

The basis of the current implementation of iTrust is the Apache Web server compiled with several PHP standard modules and library extensions. The Web server foundation component contains no custom code; all software is used as is, which enables rapid node deployment. iTrust utilizes various standard modules, including the session and logging modules described below.

The session module allows tracking of users on each node, so that multiple users can interact with the same node at the same time in a convenient manner (*i.e.*, without having to re-enter the same data on each Web page load). For example, session variables persist between multiple Web page fetches and between multiple resource retrievals. However, all session variables are purely for the convenience of the user, and a careful user may safely turn off session tracking (with only a minor inconvenience of re-entering certain data occasionally). In either case, all session data are deleted when the session (the Web browser window) is closed; there is no ability to identify a given user in subsequent sessions.

The logging module is enabled only for debugging and simulation, and can be disabled at any time by the node administrator. There is no direct relationship between the logging and session functions, *i.e.*, a user's actions cannot be tracked simply by viewing access logs (unless, of course, only one individual ever uses the node). The log file is written to disk but, optionally, may be automatically emailed to the node administrator. In the case where there are multiple nodes on the same computer, all of the nodes share the same log file and prefix each log entry with a unique node identifier.

iTrust also utilizes compiled-in modules, including cURL, SQLite, and the PHP Extension Community Library (PECL) for HTTP, as described below.

The cURL functions are used primarily for inter-node communication and resource-specific actions. When a resource is added to a node, a call may be made to that resource's URL to scan for metadata automatically. cURL automatically follows HTTP redirects and resolves file dependencies (such as HTML frame sources and image sources). Both the fetched text and the fetched images are accessible to the Java jar files, as described below.

SQLite is used for all administrative information such as node, metadata and resource information. For example, the node membership is stored in a database table, and the relationships between the metadata and the resources are stored in a normalized table. SQL constraints enforce several fundamental iTrust features, such as non-duplicate node addresses in the membership and unique resource URLs. Use of SQLite as a PHP module, instead of MySQL or PostgreSQL servers, aids with the rapid deployment of iTrust nodes. iTrust works on any reasonably modern Web host, because the file-as-a-database model of SQLite requires only minimal local write privileges.

The PHP Extension Community Library (PECL) for HTTP is an external compiled-in module used for inter-node search and metadata queries. A requesting node may use PECL HTTP to send a POST statement to a potential source node to search for the metadata that match the user's metadata query.

2.2 Application Infrastructure

The key iTrust methods reside in the application infrastructure; indeed, all of the node- and resource-related functions exist in this component. The infrastructure is divided into three parts: metadata-related functions, node- and resource-related functions, and Java jar files. All parts interact with the Web server foundation, whereas only some functions are exposed to the public interface component.

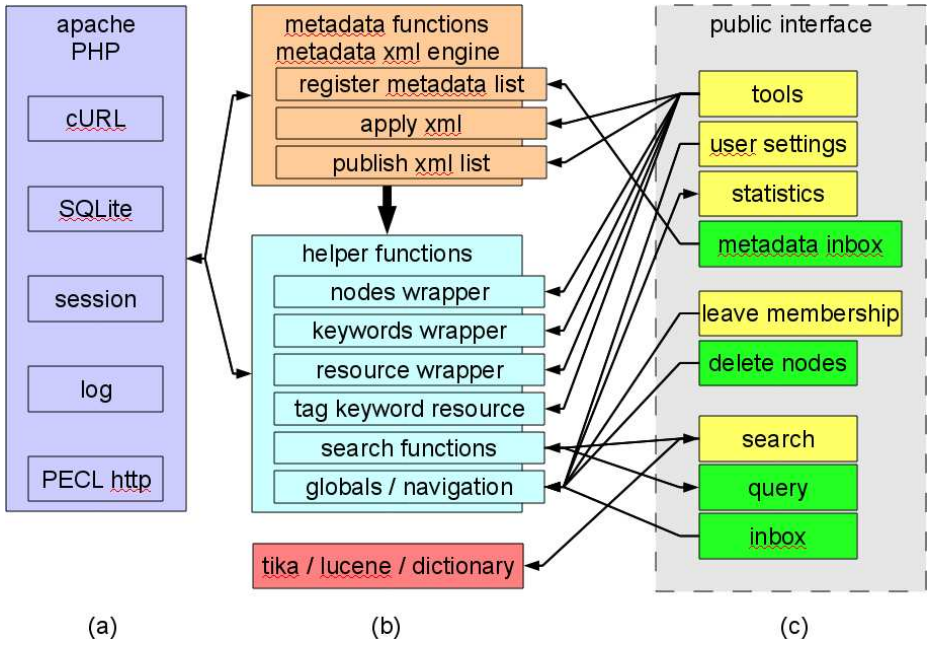

Fig. 4: The iTrust system, which comprises (a) the Web server foundation, (b) the application infrastructure, and (c) the public interface.

The creation and distribution of metadata, both internal and inter-node, are handled by the metadata-related functions. A node generates metadata from existing resources by invoking the metadata XML engine, which exhaustively scans all resources and creates an XML list describing the relationship between the metadata and the resource. Other metadata-related functions deal with the distribution of the XML list to other nodes, or with the receipt of XML lists distributed by other nodes. In the latter case, the received XML lists are scanned, and the metadata are inserted into the current node. In this way, the metadata are replicated among participating nodes.

Node- and resource-related functions, also known as helper functions, deal with bookkeeping tasks. These functions include functions that insert nodes into the membership, insert keywords into the database, and upload or fetch resources. Resources can be tagged with metadata manually by the user, or they can be automatically scanned for metadata, depending on the user's preferences. Node querying and query relaying are also handled by the helper functions (mostly through the use of PECL HTTP). All user variables (per session) and global administrative variables are stored.

Java jar files are used to generate metadata quickly and easily, and to provide the user with many conveniences. Apache's Tika and Lucene packages are used to generate metadata from resources automatically and efficiently, in the case where the user chooses not to generate metadata manually. The WordNet dictionary is used to provide the user with functions, such as spell checking and synonym suggestions.

2.3 Public Interface

The public interface, through which the users and the system administrator interact with iTrust, is divided between human and computer interfaces. Computer interfaces (dark boxes on the right in Figure 4) handle all inter-node communication such as queries, resource distribution, and metadata list distribution. All of the other interfaces (clear boxes on the right in Figure 4) are human-oriented and consist of PHP driven HTML Web pages; in fact, all human interaction with iTrust is through Web pages.

Administration is performed through the tools Web pages and other Web pages. Tools allow an administrator to add nodes or metdata keywords using simple HTML form text boxes. Adding resources requires uploading a file (form file input) or providing a URL (form text box input). User settings and statistics Web pages provide feedback to the administrator about the membership size, resource count, *etc*. An administrator may generate and distribute metadata XML lists or update the participating nodes' metadata lists. An administrator may also request that a node be removed from a node's membership. In this case, the request is activated through a human interface, and the request is distributed through the iTrust network using computer interfaces.

The most used feature of iTrust is the human interface for searching, where a user can enter a search query to request a resource. The query is sent from the current node to participating nodes using computer interfaces in a simple inbox-type fashion. Participating nodes read their inbox for queries, send back a response if there is a match, and independently decide whether to relay the query.

3. Performance Evaluation

In the performance evaluation, we consider the probability of a match, using both analysis and simulation based on our implementation of iTrust. We assume that all of the participating nodes have the same membership set. In addition, we assume that the Internet is reliable and that all of the participating nodes have enough memory to store the source files and the metadata. We randomly select nodes without repetition from the membership set for distribution of the metadata and requests. If a node receives a request and it holds the metadata that matches the metadata in the request, we say that the node has a match.

3.1 Probabilistic Analysis

First, we consider the probability that a node has a match, when all of the participating nodes are operational. Then, we consider the probability that a node has a match, when some of the participating nodes are not operational.

3.1.1 Probability that a node has a match when all of the nodes are operational

In an iTrust network with a membership of n nodes, we distribute the metadata to m nodes and the requests to r nodes. The probability p that a node has a match then is:

$$p = 1 - \frac{n-m}{n}\frac{n-1-m}{n-1}\cdots\frac{n-r+1-m}{n-r+1} \quad (1)$$

Equation (1) holds for $n \geq m+r$. If $m+r > n$, then $p = 1$. The formula is obtained as follows.

If $n \geq m+r$, first we find the probability q of no match on any of the r trials at the r nodes to which the requests are delivered. The probability of no match on the first trial is $\frac{n-m}{n}$. The probability of no match on the second trial is $\frac{n-1-m}{n-1}$, and so on. The probability of no match on the rth trial is $\frac{n-r+1-m}{n-r+1}$. Thus, the probability q of no match on any of the r trials is:

$$q = \frac{n-m}{n}\frac{n-1-m}{n-1}\cdots\frac{n-r+1-m}{n-r+1} \quad (2)$$

and the probability p of a match on one or more of the r trials is:

$$\begin{aligned} p &= 1-q \\ &= 1 - \frac{n-m}{n}\frac{n-1-m}{n-1}\cdots\frac{n-r+1-m}{n-r+1} \end{aligned} \quad (3)$$

If $m+r > n$, then the subset of nodes to which the request is delivered and the subset of nodes to which the metadata are delivered intersect in at least one node and, thus, $p = 1$.

3.1.2 Probability that a node has a match when not all of the nodes are operational

If x represents the proportion of the n nodes that are operational (and, thus, $1-x$ represents the proportion of the n nodes that are not operational), then the probability p that a node has a match is:

$$p = 1 - \frac{n-mx}{n}\frac{n-1-mx}{n-1}\cdots\frac{n-r+1-mx}{n-r+1} \quad (4)$$

Equation (4) holds for $n \geq mx+r$. If $mx+r > n$, then $p = 1$. This formula is obtained as follows.

If $n \geq mx+r$, then the probability of no match on the first trial is $\frac{n(1-x)+(n-m)x}{n} = \frac{n-mx}{n}$. The probability of no match on the second trial is $\frac{(n-1)(1-x)+(n-1-m)x}{n-1} = \frac{n-1-mx}{n-1}$, and so on. The probability of no match on the rth trial is $\frac{(n-r+1)(1-x)+(n-r+1-m)x}{n-r+1} = \frac{n-r+1-mx}{n-r+1}$. Thus, the probability q of no match on any of the r trials because all of the r nodes that receive the request are not operational or do not hold the metadata is:

$$q = \frac{n-mx}{n}\frac{n-1-mx}{n-1}\cdots\frac{n-r+1-mx}{n-r+1} \quad (5)$$

and the probability p that one or more of the r nodes that receives the request is operational and has a match is:

$$\begin{aligned} p &= 1-q \\ &= 1 - \frac{n-mx}{n}\frac{n-1-mx}{n-1}\cdots\frac{n-r+1-mx}{n-r+1} \end{aligned} \quad (6)$$

If $mx+r > n$, then the subset of nodes to which the request is delivered and the subset of nodes to which the metadata are delivered intersect in at least one node and, thus, $p = 1$.

3.2 Simulation Based on Implementation

Using our implementation of iTrust described in Section 2, we performed simulation experiments to validate Equations (1) and (4). In our simulation, we used libCURL (which is a free client-side URL transfer library for transferring data using various protocols) to collect the match probabilities.

Before we run our simulation program, we provide the following input to the program: the number n of nodes in the membership, the number m of nodes for metadata distribution, the number r of nodes for request distribution, and the proportion x of operational nodes.

First, the simulation program clears the data from the SQLite databases. Next, the program adds the nodes to the membership. Once all of the nodes are added to the membership, we call the source node to upload a file and the program then creates the corresponding metadata. Then, the simulation program randomly selects nodes for metdata distribution, and distributes the metadata to those nodes. Next, the program randomly selects the nodes for request distribution, and distributes the requests to those nodes. Then, the simulation program waits for 5 seconds. If one or more nodes has replied back to the simulation program, it means that there is a match and the program returns 1; otherwise, there is no match and the program returns 0.

We repeat the same process 100 times for the source nodes and correspondingly for the requesting nodes, and

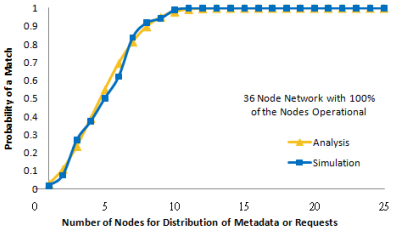

Fig. 5: Match probability vs. number of nodes for distribution of metadata and requests with 36 participating nodes, all of which are operational.

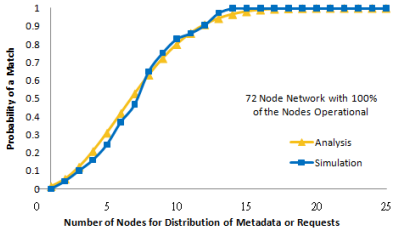

Fig. 6: Match probability vs. number of nodes for distribution of metadata and requests with 72 participating nodes, all of which are operational.

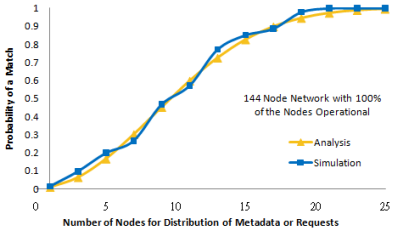

Fig. 7: Match probability vs. number of nodes for distribution of metadata and requests with 144 participating nodes, all of which are operational.

plot the mean results in our simulation graphs. We collected simulation data for 36, 72 and 144 participating nodes, when all of the nodes are operational. We also collected simulation data for 144 participating nodes when 100%, 80% and 60% of the nodes are operational.

3.3 Performance Evaluation Results

First, we consider the analytical and simulation results for the probability of a match, as the number of participating nodes increases. Then, we consider the analytical and simulation results for the probability of a match, as the proportion of non-operational nodes increases.

3.3.1 Increasing the number of participating nodes

Figures 5, 6 and 7 show both the analytical results and the simulation results for 36, 72 and 144 participating nodes, all of which are operational. The analytical curves obtained from Equation (1) are shown in the background (light curves), and the simulation curves obtained from our iTrust implementation are shown in the foreground (dark curves). We see from these figures that the simulation results are very close to the analytical results.

Figure 5 shows the match probability versus the number of nodes for distribution of metadata and requests in a network when 100% of the 36 nodes are operational. From the figure, we see that the probability increases as the number of nodes to which the metadata and requests are distributed increases. The reason is that the more nodes to which the metadata and requests are distributed, the more matches there are.

When the membership contains more nodes, the match probability asymptotically approaches 1 more slowly than for a membership with fewer nodes. That is, if we distribute the metadata and the requests to the same number of nodes, but the membership contains more nodes, the probability of a match is less than that for a membership with fewer nodes.

When we increase the membership to 72 nodes in Figure 6, the curves approach 1 more slowly than do the curves in Figure 5 for a membership containing 36 nodes. In other words, as we increase the membership, we must distribute the metadata and the requests to more nodes to obtain a higher match probability. Similarly, in Figure 7, when we increase the membership to 144 nodes, we see that the curves grow even more slowly than do the curves in the 36 node and 72 node networks.

Suppose now, for example, that we want to achieve a 0.98 match probability, in these three cases, which involve 36, 72 and 144 nodes all of which are operational. In the 36 node network, we need to distribute the metadata and the requests to only 10 nodes to achieve a 0.98 match probability. However, in the 72 node network, we need to distribute the metadata and the requests to 15 nodes to achieve a 0.98 match probability, whereas in the 144 node network, we need to distribute the metadata and the requests to 22 nodes to achieve a 0.98 match probability.

Thus, when we distribute the metadata and the requests to only a few nodes, the match probability is lower and the requester is unlikely to receive multiple responses from multiple matching nodes. When we distribute the metadata and the requests to more nodes, the match probability is higher and the requester will more likely receive multiple responses from multiple matching nodes. For a network with more participating nodes, the match probability grows more slowly than the match probability for a network with fewer participating nodes.

3.3.2 Increasing the number of non-operational nodes

Figures 8, 9 and 10 show both the analytical results and the simulation results for 144 nodes, when 100%, 80% and 60% of the participating nodes are operational, *i.e.*, when 0%, 20% and 40% of the participating nodes are non-operational. The analytical curves obtained from Equation (4) are shown in the background (light curves), and the simulation curves obtained from our iTrust implementation are shown in the foreground (dark curves). Again, we see from these figures that the simulation results are very close to the analytical results.

In Figures 8, 9 and 10, we see that the match probability curves increase as the number of nodes for distribution of metadata and requests increases. However, if we compare Figure 8 with Figure 9, we notice that the curves in Figure 8 asymptotically approach 1 faster than the curves in Figure 9. The reason is that in Figure 8 every node is operational, whereas in Figure 9 only 80% of the nodes are operational. Therefore, for distribution of metadata and requests to the

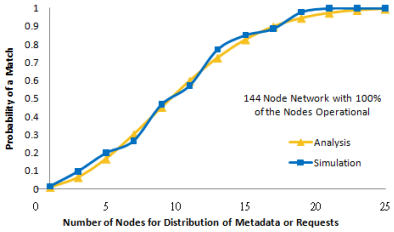

Fig. 8: Match probability vs. number of nodes for distribution of metadata and requests with 144 participating nodes where 100% of the nodes are operational.

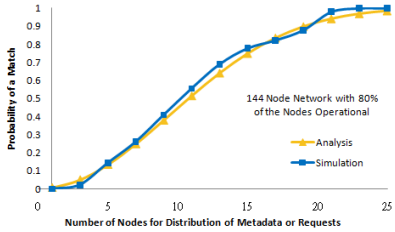

Fig. 9: Match probability vs. number of nodes for distribution of metadata and requests with 144 participating nodes where 80% of the nodes are operational.

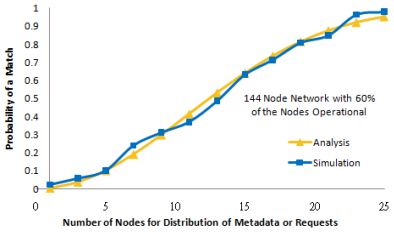

Fig. 10: Match probability vs. number of nodes for distribution of metadata and requests with 144 participating nodes where 60% of the nodes are operational.

same number of nodes, the probability of a match in Figure 8 is generally higher than it is Figure 9. Similarly, in Figure 10, where only 60% of the nodes are operational, the curves asymptotically approach 1 more slowly than the curves in Figures 8 and 9.

Suppose now, for example, that the metadata and the requests are distributed to 20 nodes, in these three cases, all of which involve 144 nodes. If 100% of the 144 nodes are operational, the probability of a match is 0.96. But, if 80% of the 144 nodes are operational, the probability of a match is 0.92, whereas if 60% of the 144 nodes are operational, the probability of a match is is 0.85, which is still quite good.

Thus, when all of the participating nodes are operational, the match probability is higher and the requester will likely receive multiple responses from multiple matching nodes. When there are fewer operational nodes, the match probability is lower and the requester is less likely to receive multiple responses from multiple matching nodes. Consequently, we must distribute the metadata and the requests to more nodes as the number of non-operational nodes increases, to obtain higher match probabilities. Nonetheless, iTrust retains significant utility even when not all of the nodes are operational, demonstrating that iTrust is quite robust.

4. Related Work

Centralized search engines for Internet search, such as Google [7], store metadata for information in a centralized index, and match queries containing keywords against the metadata at the central site. Centralized search engines are used commercially for Internet search, because they are efficient and scalable; however, they are vulnerable to manipulation by administrators. Centralized publish/subscribe systems also use a centralized index [4], against which queries are matched, raising the same issues of trust of the centralized site.

Risson and Moors [14] provide a survey of search in peer-to-peer networks, and Mischke and Stiller [12] provide a taxonomy of distributed search in such networks. Distributed publish/subscribe strategies are categorized as either structured based on managed overlay networks, or as unstructured based on gossiping and randomization. The structured approach is more efficient than the unstructured approach, but it involves administrative control with a consequent risk of manipulation. iTrust falls within the unstructured distributed search category.

Gnutella [8], one of the first unstructured networks, uses flooding of requests to find information. An extension of Gnutella involves supernodes [19], which improves efficiency but incurs some of the trust risks of centralized strategies. Freenet [2] is more sophisticated and efficient than Gnutella, because it learns from previous requests. In Freenet, nodes that successfully respond to requests subsequently receive more metadata and more requests. Thus, it is easy for a group of untrustworthy nodes to conspire together to gather most of the searches into their group, rendering Freenet vulnerable to subversion.

Sarshar *et al.* [15] combine random walks and data replication with a two-phase query scheme in a Gnutella-like network. BubbleStorm [16] replicates both queries and data, and combines random walks with flooding. GIA [1] combines biased random walks with one-hop data replication. Lv *et al.* [10] show that path replication and random replication are near-optimal in unstructured peer-to-peer networks. Like these systems, iTrust exploits randomization and replication.

Ferreira *et al.* [6] use a random-walk strategy to replicate both queries and data to the square root of the number of nodes in the network. Zhong and Shen [20] use random walks for requests, where the number of nodes visited by a request is proportional to the square root of the request popularity. Cooper [3] exploits search trees whose node degrees approximate the square root of the size of the network. Like these researchers, we can also exploit the square root function in iTrust.

Pub-2-Sub [17] is a publish/subscribe service for unstructured peer-to-peer networks of cooperative nodes. Instead of gossiping, Pub-2-Sub uses directed routing to distribute subscription and publication messages to the nodes. None of the above unstructured systems is particularly concerned with trust, as is iTrust. Rather, their objective is efficiency, which is not the primary concern of iTrust.

Systems for social networks [11], [13] exploit the trust that members have in other members, and route information

and requests based on relationships among members. Social networks, like Facebook [5], are centrally administered and, thus, depend on benign administrators.

There exist a few systems for social networks that, like iTrust, are concerned with trust. OneSwarm [9] is a peer-to-peer system that allows data to be shared either publicly or anonymously, using a combination of trusted and untrusted nodes. OneSwarm is part of an effort to provide an alternative to cloud computing that does not depend on centralized trust. Its initial goal is to protect the privacy of the users, which iTrust does not aim to do. Quasar [18] is a probabilistic publish/subscribe system for social networks. The authors note that "an unwarranted amount of trust is placed on these centralized systems to not reveal or take advantage of sensitive information." Thus, the trust objective of Quasar is quite different from that of iTrust.

5. Conclusions and Future Work

We have described the iTrust system, a distributed search and retrieval system for the Internet with no centralized mechanisms and no centralized control. iTrust is particularly valuable for individuals who fear that the conventional centralized Internet search mechanisms might be subverted or censored. The very existence of iTrust can help to deter attempts to subvert the conventional Internet search mechanisms, and can provide assurances to individuals that the information they seek will be available to them.

In the future, we plan to evaluate the effectiveness, efficiency, scalability, and reliability of the iTrust system in PlanetLab. In addition, we plan to conduct further probabilistic analyses and to investigate a range of possible attacks on iTrust and countermeasures to such attacks. Our objective for iTrust is a network in which individual nodes can detect a potential attack, and can adapt to an attack to maintain trustworthy distributed search and retrieval even when the network is under attack.

Acknowledgment

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

References

[1] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham and S. Shenker, "Making Gnutella-like P2P systems scalable," *Proceedings of the ACM SIGCOMM Applications Technologies, Architectures and Protocols for Computer Communications Conference*, Karlsruhe, Germany, August 2003, pp. 407–418.

[2] I. Clarke, O. Sandberg, B. Wiley and T. Hong, "Freenet: A distributed anonymous information storage and retrieval system," *Proceedings of the Workshop on Design Issues in Anonymity and Unobservability*, Lecture Notes in Computer Science, Berkeley, CA, July 2000, pp. 46–66.

[3] B. F. Cooper, "Quickly routing searches without having to move content," *Proceedings of the 4th International Workshop on Peer-to-Peer Systems*, Lecture Notes in Computer Science 3640, Ithaca, NY, February 2005, pp. 163–172.

[4] P. T. Eugster, P. A. Felber, R. Guerraoui and A. M. Kermarrec, "The many faces of publish/subscribe," *ACM Computing Surveys* 35:2, June 2003, pp. 114–131.

[5] Facebook, http://www.facebook.com

[6] R. A. Ferreira, M. K. Ramanathan, A. Awan, A. Grama and S. Jagannathan, "Search with probabilistic guarantees in unstructured peer-to-peer networks," *Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing*, Konstanz, Germany, August 2005, pp. 165–172.

[7] Google, http://www.google.com

[8] Gnutella, http://gnutella.wego.com/

[9] T. Isdal, M. Piatek, A. Krishnamurthy and T. Anderson, "Privacy preserving P2P data sharing with OneSwarm," Technical Report UW-CSE, Department of Computer Science, University of Washington, 2009.

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

[11] S. Marti, P. Ganesan and H. Garcia-Molina, "SPROUT: P2P routing with social networks," *Proceedings of Current Trends in Database Technology Workshop*, Lecture Notes in Computer Science 3268, November 2004, pp. 425–435.

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

[13] A. Mislove, K. P. Gummadi and P. Druschel, "Exploiting social networks for Internet search," *Proceedings of the 5th Workshop on Hot Topics in Networks*, Irvine, CA, November 2006.

[14] J. Risson and T. Moors, "Survey of research towards robust peer-to-peer networks: Search methods," Technical Report UNSW-EE-P2P-1-1, University of New South Wales, September 2007, RFC 4981, http://tools.ietf.org/html/rfc4981

[15] N. Sarshar, P. O. Boykin and V. P. Roychowdhury, "Percolation search in power law networks: Making unstructured peer-to-peer networks scalable," *Proceedings of the 4th International Conference on Peer-to-Peer Computing*, Zurich, Switzerland, August 2004, pp. 2–9.

[16] W. W. Terpstra, J. Kangasharju, C. Leng and A. P. Buchman, "BubbleStorm: Resilient, probabilistic, and exhaustive peer-to-peer search," *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures and Protocols for Computer Communications*, Kyoto, Japan, August 2007, pp. 49–60.

[17] D. A. Tran and C. Pham, "Enabling content-based publish/subscribe services in cooperative P2P networks," *Computer Networks* 52:11, August 2010, pp. 1739-1749.

[18] B. Wong and S. Guha, "Quasar: A probabilistic publish-subscribe system for social networks," *Proceedings of the 7th International Workshop on Peer-to-Peer Systems*, Tampa Bay, FL, February 2008.

[19] B. Yang and H. Garcia-Molina, "Improving search in peer-to-peer networks," *Proceedings of the 22nd IEEE International Conference on Distributed Computing Systems*, Vienna, Austria, July 2002, pp. 5–14.

[20] M. Zhong and K. Shen, "Popularity-biased random walks for peer-to-peer search under the square-root principle," *Proceedings of the 5th International Workshop on Peer-to-Peer Systems*, Lecture Notes in Computer Science 4490, 2006.