6S: DISTRIBUTING CRAWLING AND SEARCHING ACROSS WEB PEERS

Le-Shin Wu
Computer Science Dept.
Indiana University
Bloomington, IN 47405
lewu@cs.indiana.edu

Ruj Akavipat
Computer Science Dept.
Indiana University
Bloomington, IN 47405
rakavipa@cs.indiana.edu

Filippo Menczer
School of Informatics
& Computer Science Dept.
Indiana University
Bloomington, IN 47405
fil@indiana.edu

ABSTRACT
A collaborative peer network application called 6Search (6S) is proposed to address the scalability limitations of centralized search engines. 6S peers depend on a local adaptive routing algorithm to dynamically change the topology of the peer network and search for the best neighbors to answer their queries. We validate prototypes of the 6S network via simulations with $70 - 500$ model users based on actual Web crawls and find that the network topology rapidly converges from a random network to a small world network, with clusters emerging from user communities with shared interests. We finally compare the quality of the results with those obtained by centralized search engines such as Google, suggesting that 6S can draw advantages from the context and coverage of the peer collective.

KEY WORDS
Peer collaborative search, Web information retrieval

1 Introduction and Background

Centralized search engines have difficulty with coverage of the Web [6] because the Web is large, fast-growing and fast-changing [3, 4, 8]. Further, various biases introduced to address the needs of the "average" user imply diminished effectiveness in satisfying many atypical search needs. We identify the above limitations as problems of *scale*.

It is evident that distributed systems are part of the answer to the scale problem and peer network is increasingly seen as a candidate framework for distributed Web search applications. One model proposed by the YouSearch project is to maintain a centralized search registry for query routing (similar to Napster) , and moreover enrich the peers with the capability to crawl and index local portions of the Web [2]. The central control in this approach unfortunately makes it difficult to adapt the search process to the heterogeneous and dynamic contexts of the peer users.

The opposite, completely decentralized approach (as in early versions of Gnutella) is another option. But this approach has the well-known disadvantage that peers are flooded with traffic from queries and responses. To regulate the traffic, adaptive content based routing has been proposed for the file sharing setting [5] where the routing mechanism relies on metadata describing nodes' contents.

Combined approaches between the above two are also available such as distributed routing tables [10, 11] and hierarchical routing between hub and leaf peers [7].

In this paper we propose a different model for peer-based Web search, which uses the idea of modeling neighbor nodes by their content but without assuming the presence of special directory hubs. Each peer is both a (limited) directory hub and a content provider; it has its own topical crawler and local search engine. Queries are first matched against the local engine, and then routed to neighbor peers to obtain more results. Initially the network has a random topology but our protocol includes a learning algorithm by which each peer uses the results of its interactions with neighbors (matches between queries and responses) to refine its model of the other peers.

The peer network should lead to the emergence of a clustered topology by intelligent collaboration between the peers. In fact, we predict that the ideal topology for such a network would be a "small world" [14]. Following Milgram's famous experiments on "six degrees of separation" [14], we named our model *6Search* (6S).

After a brief introduction of the 6S protocol, architecture, and adaptive query routing algorithm, the paper presents the results of extensive experiments based on model users and simulated queries using real, distributed Web crawl data.

2 6S Protocol and Algorithms

The 6S protocol layer sits between the application and the network (TCP/IP) stack. The design of our protocol is based on the following considerations: (1) peers are independent; (2) a peer can enter and leave the network at any time; (3) a peer should not be overwhelmed by other peers; (4) a query should not be propagated indefinitely; (5) a peer may choose not to respond or forward some queries; and (6) the architecture should make it difficult to create denial of service (DoS) attacks using the service.

2.1 Message Primitives

There are four major message primitives on which 6S prototype protocol is based. (1) The *query message* consists of query keywords, weight of each keyword, `ID`,

`timestamp`, and the `TTL` (time to live). The purpose of the `TTL` is to limit propagation of a query in the network by decreasing the value of the TTL's counter each time the query is forwarded until the counter value is 0. (2) The *query response* is used by a peer to respond to other peers' search queries. The response must be sent back to the neighbor from which we received the query regardless of which peer originated the query. This is to prevent DoS attacks created by exploiting the response system. (3) The *profile request* allows a peer to request profiles from others. The profile describes what a peer has indexed and is ready to share.We use the top N most frequent keywords in a peer's index as the initial profile. (4) The *profile response* allows a peer to respond to a request for its own profile. Section 2.3 describes how a profile is generated and updated by a peer.

2.2 Neighbor Management in 6S Networks

To form communities without centralized control or aggressive flooding of the network, the 6S peer needs to find new peers through its current neighbors. The approach that we use in our prototype is to let a peer attach its contact information, a unique ID (`ownerid`), with the query. If the peer that receives a query wants to become a neighbor of the requesting peer, it will add its own contact `ID` into the query response message. The new neighbor peers can later contact each other directly.

The 6S protocol gives each peer a fixed number of slots for neighbors, N_n, depending on their bandwidth and computational power to process neighbor data. A peer will search for new peers when its neighbor slots are not full or when it wants to find better neighbors than the currently known peers. Each peer may of course know about more than N_n peers but the maximum number N_k of known peers will be capped by the peer application's available memory or storage.

Many query routing algorithms in the P2P literature (cf. Section 1) require peers to send update messages in order to maintain valid network information when peers leave the network. In contrast, a 6S peer does not need to send any messages when it wants to leave the network because our routing algorithm (described in the following section) dynamically updates the neighbor profiles based on queries and responses in the system.

2.3 Adaptive Query Routing

Each peer will learn and store profiles of other peers to support adaptive query routing. A neighbor profile is the information a particular peer maintains to describe its knowledge about what that neighbor stores in its search engine index. By using profile information, peers try to increase the probability of choosing the appropriate neighbors to route their queries.

In our first 6S prototype, we implemented a very simple method to initialize and maintain peer profiles consisting of two steps. First, a peer asks a neighbor for its description, defined as a list of n most frequent keywords in the neighbor's index. Second, the peer performs a crude update to this list by adding query terms for which the neighbor returns good responses. The score of a keyword in such a neighbor profile is the highest similarity score of the responses a neighbor returns for that keyword. This method was shown to give rise to an efficient network topology and promising initial results [1]. In this paper, we improve the reliability and robustness of the simple learning algorithms discussed above by introducing a better profile representation and a novel soft updating scheme.

Interactions with peers reveal information of varying reliability. For example, a direct response to a query is telling about a peer's knowledge with respect to that query, but may also reveal (less reliable) information about the peer's knowledge relative to other queries. We want to capture all available information in profiles, but must discriminate information on the bases of its reliability. We let each peer maintain two profile matrices, W^f and W^e for *focused* and *expanded* information, respectively. Each profile matrix has the same structure and is initially empty; rows correspond to terms and columns to peers. Thus an element $w_{i,p}$ of W is the contribution of term i to the profile of known peer p ($p = 1, \ldots, N_k$).

Focused profile: weights $w^f_{i,p}$ are initially updated based on p's response to a neighbor profile request (cf. Section 2.1), and successively updated through query-response interaction—namely for terms i in queries submitted or forwarded to p. Based on the comparison of the incoming hits with its local hits for a query Q, the peer makes an assessment about p's knowledge with respect to terms $i \in Q$.

Expanded profile: weights $w^e_{j,p}$ are updated through query-response interaction analogously to the focused profile, but for terms $j \notin Q$ that co-occur with terms $i \in Q$ in a hit page d returned by p, such that j has a higher term frequency: $TF(j,d) > \max_{i \in Q} TF(i,d)$. If a certain set of documents is a good response for a certain query, then it may as well be a good response for queries that are well represented in the set. By this query expansion, we expect to speed up neighbor learning since queries are typically short and thus W^f is typically rather sparse.

The 6S peer updates its neighbor profile when it gets a query response. Here we propose the following learning rule, which we will refer to as *soft updating*, to modify the weights of the query terms in the neighbor profile matrices:

$$w_{i,p}(t+1) = (1-\gamma) \cdot w_{i,p}(t) + \gamma \cdot \frac{S_p + 1}{S_l + 1} \qquad (1)$$

where t is a time step, S_p and S_l are the average scores of p's hits and the local hits respectively in response to the

query Q, and γ is a learning rate parameter ($0 < \gamma < 1$). The terms i subject to this learning rule depend on Q and the profile matrix (focused or expanded) as described above.

The actual set of N_n neighbors, i.e. those to whom queries are sent, is selected dynamically for each query at time t among the $N_k(t)$ known peers. The adaptive routing algorithm to manage neighbor information and to use such information for dynamically selecting neighbors to a query is described by the following steps:

1. Request a profile from a new peer when it is first discovered. Next, initialize a description for the peer using the list of keywords contained in the peer's profile.

2. Evaluate responses from neighbors (and neighbors' neighbors, and so on) to query Q to update the description of each known peer: (a) Compute S_p and S_l. (b) Update W^f using Equation 1 for terms in Q. (c) If $S_p > S_l$ update W^e using Equation 1 for terms not in Q that occur in the hits received from neighbors more frequently than the query terms. (d) Send the discovery signal (`ownerid`) with the next query to that neighbor. (e) Add new peers that respond to discovery signals to the list of known peers with their corresponding profiles.

3. To route the next query Q', rank known peers by similarity σ computed as follows:

$$\sigma(p, Q') = \sum_{i \in Q'} \left[\alpha \cdot w^f_{i,p} + (1 - \alpha) \cdot w^e_{i,p} \right] \quad (2)$$

   where α is a reliability parameter, typically $0.5 < \alpha < 1$ to reflect higher confidence in focused profile weights as they come from direct responses to queries. Select the top N_n ranked known peers as neighbors, and send/forward Q'.

4. Goto step 2.

3 Experimental Setup

We created two different types of simulators that allow us to model artificial users and run their queries over real indexes obtained from actual distributed Web crawls. Our simulators take a snapshot of the network for every time step. In a single time step of the simulator, all of the peers process all of their buffered incoming messages and send all of their buffered outgoing messages.

The first simulation models a relatively small network with relatively large peers while the second models a larger network with relatively light-weight peers. Specifically, there are $N = 70$ and $N = 500$ peers belonging to 7 and 50 different groups of 10 peers each, in our first and second simulation, respectively. Each group is associated with a general topic. Each peer has its own topical crawler and peer search engine, but for the peers in a given group the search engines are built by topical crawlers focusing on the same topic. For example, if a group's topic is "sports," then all the peer search engines in this group focus on different aspects of sports. Two points are to be emphasized here. First, 6S peers can have multiple and broad topics even though our experiment limits peers to have only relatively narrow topics. Second, while we simulate these communities to see if the peer network can discover them, any individual peer has no more knowledge about other. peers in its group than about all other peers.

Group topics are chosen from the Open Directory[1] (ODP) to simulate the group structure, according to a simple methodology developed to evaluate topical crawlers [9]. For example, `Computers/Software/Freeware` is one group topic in our first simulation. For each group, we extract a set of 100–200 URLs from the ODP subtree rooted at the category node corresponding to the group's topic. Random subsets are assigned to the peer crawlers as seeds. So the search engines within each group differ from each other according to the different sets of crawled pages. The peer uses its group topic and its own seed URLs to crawl 10,000 pages in our first simulation and 1,000 pages in our second simulation. Each peer is allowed to know about all of the other peers ($N_k = 69$ for first simulation and $N_k = 499$ for second simulation) and to have $N_n = 5$ neighbors. At the beginning of each experiment, the peer network is initialized as a random *Erdos-Renyi* graph, i.e., each peer is assigned $N_n = 5$ random neighbors drawn from a uniform distribution, irrespective of groups.

Each peer in our experiments has 10 local 3–5 words queries. For the first simulation, the queries for each peer were generated by randomly picking keywords from the ODP descriptions of the Web sites whose URLs were used as seeds for the peer's crawler. For the second simulation, instead of ODP descriptions we use the title strings of the Web sites. The peer that has a local query from a certain Web site and the peer that used the URL of this Web site as a seed for the peer's crawler belong to the same group.

Finally we empirically set the profile learning rate to $\gamma = 0.3$ (Equation 1), the profile reliability parameter to $\alpha = 0.8$ (Equation 2), and the `TTL` to 3. We ran the simulator for about 10,000 time steps for the first simulation (corresponding to 1,000 queries issued per peer) and 1,200 time steps for the second simulation (corresponding to 120 queries issued per peer).

Our first experiment was performed on IU's AVIDD Linux cluster with 208 2.4 GHz Prestonia processors. A complete simulation run took approximately 6 hours. Our second experiment was distributed over 5 dual 2.8 GHz Linux machines, each running 100 peers. A complete simulation run took approximately 24 hours.

4 Analysis for Few Large Peers

With the purpose of showing the variation of the network topology at different simulation time steps, we need to in-

[1] `http://dmoz.org`

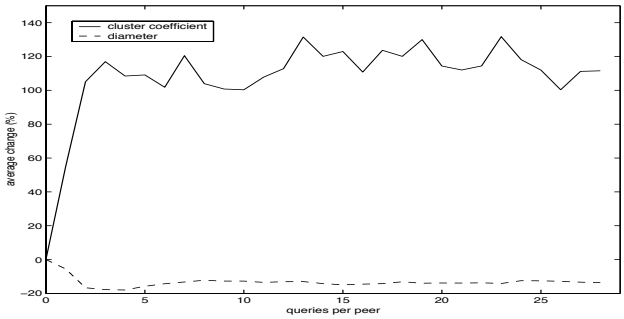

Figure 1. Small world statistics of the 6S peer network.

troduce two network statistics, the *cluster coefficient* and the *diameter*. The *cluster coefficient* for a node is the fraction of a node's neighbors that are also neighbors of each other. This is computed in the directed graph based on each peer's N_n neighbors. Thus, in our simulation, with $N_n = 5$, the total number of possible directed links between neighbors is $N_n(N_n - 1) = 20$. The overall cluster coefficient C is computed by averaging across all peer nodes. The *diameter* is defined as the average shortest path length ℓ across all pairs p of nodes. We compute the average shortest path as $D = (\frac{N}{N-1} \sum_{ij} \ell_{ij}^{-1})^{-1}$, a measure that is robust with respect to the fact that the network is not necessarily strongly connected, and therefore some pairs do not have a directed path ($\ell = \infty$). Figure 1 shows that the 6S diameter remains roughly equal to the initial random graph diameter, while the cluster coefficient increases very rapidly and significantly, stabilizing around a value twice as large as that of the initial random graph after only 5 queries per peer. These conditions define the emergence of a small world topology in our peer network [14].

To illustrate the small world phenomenon, Figure 2 shows the transformations of the peer network topology for the whole network and the neighborhood of a single group. One can observe that there are more local (within group) links and fewer long (cross-group) links on the right-hand-side, revealing the emergence of local clusters in the network topology as the semantic locality is discovered among peers.

We want to evaluate the quality of results obtained through 6S, and compare them to the results obtained from a centralized search engine using the same amount of network resources as a 6S run. For this purpose, we plot precision versus recall, a standard technique in information retrieval. For comparison, we build a centralized search engine by crawling and indexing 700,000 pages from the same seeds but using a traditional (breadth-first) crawler rather than a topical crawler. We issue the same queries used for 6S and collect 100 top hits for each query. To capture the users' relevance contexts, we extend each of the 700 peer queries with a single most frequent term from the profile of the peer owning the query. Each extended query is submitted to a separate, centralized search engine that combines the 70 peers' search engine databases; the top 100 hits returned are used as the relevant set of each query.

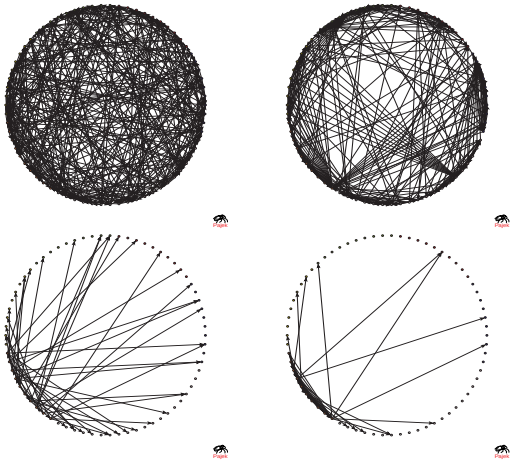

Figure 2. Peer network connectivity for all groups (top) and for one of the groups (bottom). Left: initial neighbor links. Right: final neighbor links.

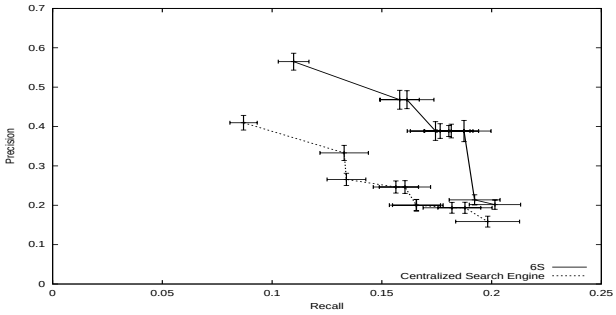

Figure 3. Precision-recall plots for 6S and the centralized search engine. Error bars correspond to standard errors of precision and recall averaged across queries.

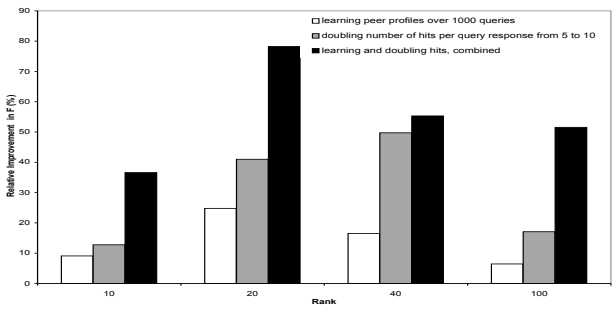

Figure 4. Relative improvement in F-measure due to query routing based on simple learning and to increasing the number of hits per query response N_h from 5 to 10, plotted versus the total number of top hits considered. The gray bars measurements are taken at the beginning of the simulation.

Figure 3 shows the precision-recall plots comparing quality of results by 6S and the centralized search engine. 6S significantly outperforms the centralized search engine. This occurs because queries are successfully routed to those peers who can return highly relevant hits due to their stronger focus relative to user interests.

Figure 4 shows that performance improves as peers learn to route queries to the appropriate neighbors, and

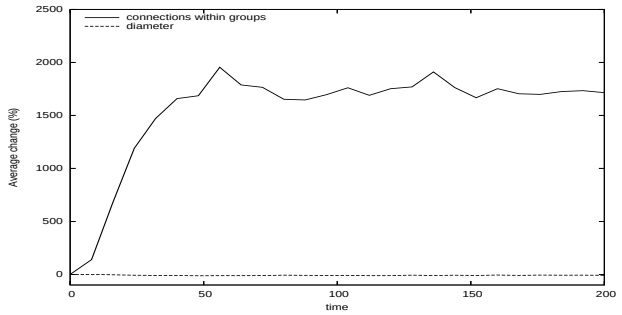

Figure 5. Average fraction of neighbors that are connected to peers in the same group as themselves, and diameter of peer network versus time.

as the number of hits N_h that peers return in response to queries increases. Performance is measured by the F-measure, which combines precision and recall through their harmonic mean. The relatively small improvement due to the simple learning algorithm motivated our design of more sophisticated adaptive query routing schemes.

5 Analysis for Many Small Peers

For our second simulation, we set the number of hits per query response $N_h = 10$. In order to evaluate the query routing algorithm described in Section 2.3, here we use two baseline query routing algorithms that do not employ the expanded profile W^e: (1) *simple* updates W^f by replacing $w^f_{i,p}$ with the best hit score from p; (2) *soft update* uses W^f with the update rule in Equation 1.

Figure 5 shows that the average fraction of neighbors that are in the same interest group as a peer increases significantly and rapidly with time. This indicates that with the proposed adaptive query routing algorithm, even though the ratio of related peers is smaller compared to the first simulation ($1/50$ rather than $1/7$), a peer can still quickly find other peers with similar interest focus.

The relevant sets in the second simulation are simply the sets of URLs classified by the ODP under the same topic as the page whose title is used as query. We show precision-recall snapshots in Figure 6. Already at the start, we observe a difference in performance between the learning algorithms. However, during the 4 time steps, the first query took to propagate (it can only travel as far as half the round trip) adaptive peers in the query path had already learned about their neighbors, hence they could better forward the query. Besides showing that all query routing schemes take advantage of the learning and improve their performance over time, Figure 6 also confirms that the more sophisticated learning algorithm outperforms the simpler ones, with the best performance achieved by combining expanded profiles and the soft profile update rule.

After all, we compare the quality of the average results obtained by our model with those returned by a real-world search engine. To this end, we queried the Google Web API. As a summary performance measure, we employed the commonly used average *precision at 10*, $\langle P_{10} \rangle$.

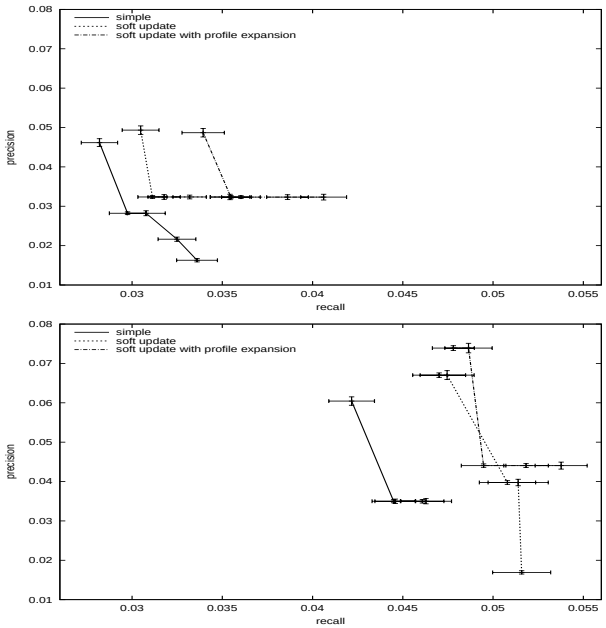

Figure 6. Precision-recall plots for three learning schemes, taken at the start of the simulation (top), and after 1000 time steps or 125 queries per peer (bottom).

Table 1. Average precision at 10 of Google and 6S.

	$\langle P_{10} \rangle$	$\sigma_{\langle P_{10} \rangle}$	95% Conf. Interval
Google	0.079678	0.00095	(0.0778, 0.0816)
6S	0.078380	0.00062	(0.07714,0.07962)

As shown in Table 1, the difference in performance between the two systems is not statistically significant, suggesting that our model can be competitive with much larger search engines — Google knows about 10^4 more pages than the entire 6S collective. The pages used as relevant sets in this experiment (ODP pages) are well known to Google, and using their titles as queries allowed Google to retrieve and rank very highly the pages with those titles. However, our model users can exploit their context and share their knowledge via collaboration during the search process, while Google has a single, universal ranking function and cannot exploit such context. Thus, Google did not rank as highly pages that our model users considered relevant because highly related to the page used to compose the query. Another factor to be considered is that Google may have returned other relevant pages which were not in our relevant sets; our automatic assessment methodology would not allow us to give credit for those. Despite this caveat, we find the comparative results very encouraging.

6 Conclusion

In this paper, we introduced a collaborative peer network application called 6Search, with which we intend to study the idea that the scalability limitations of centralized search engines can be overcome via distributed Web crawling and searching methodology. The results presented here seem to

support the idea that adaptive routing can work with real data and that critical network structure can emerge spontaneously from the local interactions between peers, capturing the locality of content interests among them. Our experiments also suggest that 6Search can outperform centralized search engines, which cannot take advantage of user context in their crawling and searching processes.

One can observe a sharp drop in precision as recall increases (Figures 3 and 6), which corresponds to the drop in F-measure as each peer considers more hits (Figure 4). The reason is that each neighbor contributes a small number N_h of hits, so in order to increase recall a peer must consider a larger pool of neighbors, some of which may belong to different topical communities.

As a project in its infancy stage, 6S has many directions for further development. One technique that we intend to test for Web searching is query relaxation which was proposed in a semantic Web setting where peers query for RDF data [12]. A robust algorithm is to be developed for combining hits from peers in the Combinator Module, thus allowing for heterogeneous scoring by peer search engines. We also plan to study the use of reinforcement learning algorithms for identifying good neighbors not only from their individual performance but also that of their neighborhoods. Finally the network security issue needs to be elaborated in our future implementation.

In parallel with the above algorithmic extensions, implementation of a working 6S peer application is under way. We are developing a prototype based on the JXTA framework [13], which will integrate the 6S protocol, topical crawler, document index system, search engine system, bootstrap system, and network communication system; we plan to release the prototype to the open-source community. Testing the prototype in a realistic setting will allow us to tune our protocols and algorithms. For example, while a peer may decide not to share its knowledge with other peers, we will consider whether the information available to a peer should be dependent on the information it is willing to share. Testing under realistic conditions will also allow us to study the robustness and scalability of the network under large scale usage.

7 Acknowledgments

We are grateful to the Nutch Organization for its open source search engine code, and to the ODP for the data used to model our simulated uses. We thank Thomas R. Reichherzer for his careful reading. This work was supported in part by NSF Career Grant IIS-0348940.

References

[1] R. Akavipat, L.-S. Wu, and F. Menczer. Small world peer networks in distributed Web search. In *Alt. Track Papers and Posters Proc. 13th International World Wide Web Conference*, pages 396–397, 2004.

[2] M. Bawa, R. Bayardo Jr, S. Rajagoplan, and E. Shekita. Make it fresh, make it quick — searching a network of personal webservers. In *Proc. 12th International World Wide Web Conference*, 2003.

[3] B. E. Brewington and G. Cybenko. How dynamic is the Web? In *Proc. 9th International World-Wide Web Conference*, 2000.

[4] D. Fetterly, M. Manasse, M. Najork, and J. Wiener. A large-scale study of the evolution of Web pages. In *Proc. 12th International World Wide Web Conference*, 2003.

[5] S. Joseph. Neurogrid: Semantically routing queries in Peer-to-Peer networks. In *Proc. Intl. Workshop on Peer-to-Peer Computing*, 2002.

[6] S. Lawrence and C. Giles. Accessibility of information on the Web. *Nature*, 400:107–109, 1999.

[7] J. Lu and J. Callan. Content-based retrieval in hybrid peer-to-peer networks. In *Proc. 12th Intl. Conf. on Information and Knowledge Management (CIKM'03)*, 2003.

[8] A. Ntoulas, J. Cho, and C. Olston. What's new on the Web?: The evolution of the Web from a search engine perspective. In *Proceedings of the 13th international conference on World Wide Web*, pages 1–12. ACM Press, 2004.

[9] P. Srinivasan, G. Pant, and F. Menczer. A general evaluation framework for topical crawlers. *Information Retrieval*, 8, 2005. Forthcoming.

[10] C. Suel, T amd Mathur, J.-W. Wu, J. Zhang, A. Delis, M. Kharrazi, X. Long, and K. Shanmugasundaram. ODISSEA: A Peer-to-Peer architecture for scalable Web search and information retrieval. In *International Workshop on the Web and Databases (WebDB)*, 2003.

[11] C. Tang, Z. Xu, and S. Dwarkadas. Peer-to-peer information retrieval using self-organizing semantic overlay networks. In *Proc. ACM SIGCOMM '03*, 2003.

[12] C. Tempich, S. Staab, and A. Wranik. REMINDIN': Semantic query routing in peer-to-peer networks based on social metaphors. In *Proc. 13th conference on World Wide Web*, pages 640–649. ACM Press, 2004.

[13] S. Waterhouse. JXTA Search: Distributed search for distributed networks. Technical report, Sun Microsystems Inc., 2001.

[14] D. Watts and S. Strogatz. Collective dynamics of "small-world" networks. *Nature*, 393:440–442, 1998.