Studying search networks with SIL

Brian F. Cooper and Hector Garcia-Molina

Department of Computer Science
Stanford University
{cooperb,hector}@db.stanford.edu

Abstract

We present a general model, called the Search/Index Link (SIL) model, for studying peer-to-peer search networks. This model allows us to analyze and visualize existing network architectures. It also allows us to discover novel architectures that have desirable properties. Finally, it can be used as a starting point for developing new network construction techniques.

1 Introduction

The recent explosion in popularity of peer-to-peer search networks has sparked great interest in the problem of designing "good" networks. Although P2P systems hold the promise of harnessing large numbers of distributed resources, so far it has been difficult to achieve scalability. Similarly, the autonomy of peers and redundancy of links should enhance fault tolerance, and yet many networks are vulnerable if failures occur at the wrong place in the network.

Our approach to dealing with these issues is to construct a general model, called the Search/Index Link (SIL) model, for studying alternative architectures for peer-to-peer search networks. The SIL model is useful for visualizing as well as analyzing existing search networks in terms of scalability, fault tolerance and other metrics. However, the simplicity and generality of the model also makes it useful for discovering new types of networks that have desirable properties. Using the SIL model, we can examine the inherent properties of an existing or new network topology, and tune the basic architecture for a given goal, such as reduced load. We have also begun to use the insights gained from the SIL model to study "ad hoc, self-supervising networks," or networks where nodes simply make and break links at will, and the networks evolves and becomes increasingly efficient over time.

The focus of the SIL model is on flooding-based networks (such as Gnutella or supernode networks), and not necessarily routing-based networks (such as distributed hash tables or DHTs [11, 10]). Although DHTs have significant advantages in many situations, we feel that the potential of flooding-based networks has not been fully realized, and that there are still many interesting research questions about flooding networks. SIL is an attempt to see what else is possible with flooding networks, beyond what has been developed so far. Moreover, flooding networks seem especially adept at content-discovery, in contrast to DHTs, whose main strength is in file location once a name is known (as pointed out in [10]. Finally, flooding networks continue to have huge popularity and wide deployment; on a typical day, Kazaa supports several million simultaneous users, and allows them to search for hundreds of millions of files and multiple petabytes of data. Optimizing such widely used systems continues to be an important research challenge. Although we could generalize SIL to model DHTs as well, the simplicity of the model as it is now gives us great power to describe and analyze P2P search networks.

In this position paper, we describe the SIL model, illustrate its usefulness for discovering novel topologies and developing network construction techniques, and discuss interesting research challenges.

2 The Search/Index Link model

A primary goal of a peer-to-peer search network is to allow member peers to search for and retrieve content stored at other peers. Searching is usually accomplished by sending queries to peers, and these peers respond with search results if they have content matching the query. Often, indexing is employed to enhance scalability and efficiency. Indexing can be used to allow a single peer to answer queries for multiple other peers without the need for those other peers to process the queries themselves. Indexing can also improve availability, since a peer can be searched even if it is temporarily unreachable.

The Search/Index Link (SIL) model generalizes the basic techniques of querying and indexing in a search network. SIL models a peer-to-peer search network overlay as a set of nodes in a graph with specialized links connecting the nodes. There are four kinds of directed links in the model:

- A *forwarding search link* (FSL) carries search messages from peer X to peer Y. Peer Y processes the query and also forwards it on outgoing FSLs. FSLs can be graphically represented as $X \Longrightarrow Y$.
- A *non-forwarding search link* (NSL) carries search messages from peer X to peer Y. Peer Y processes the query but does not forward it. NSLs are represented as $X \longrightarrow Y$.
- A *forwarding index link* (FIL) carries index updates from peer X to peer Y. These index updates inform Y about new, modified or deleted content at peer X. Peer Y integrates the index updates into its own index, and also forwards the updates on outoing FILs. FILs are represented as $X \Rrightarrow Y$.
- A *non-forwarding index link* carries index updates from X to Y. Peer Y should add the updates to its own index but does not forward them. NILs are represented as $X \dashrightarrow Y$.

Under this model, there are two basic types of activity: indexing and searching. Peers construct indexes over their own content to assist in answering searches. These indexes may be inverted lists of words, sets of metadata or simply a list of filenames. When a peer receives a search, that peer searches its index for matching documents and returns any matches as search results. Whenever a peer A updates its own index, it should also send those index updates along outgoing index links. The peers that receive these updates will effectively have a copy of A's index, and can perform searches over A's content just as well as A can. Note that these peers do not store a replica of A's content, but only index entries that aid in searching that content. For example, imagine a peer B that has a copy of A's index. When B receives a search message, B processes that search over its own content as well as over A's index, and returns search results for matches at either A or B.

We say that a node X searches a node Y *directly* by sending searches to Y, or that X searches Y *indirectly*, by sending searches to a node Z that has a copy of Y's index. The total number of nodes that X can search directly or indirectly is X's *coverage*.

The SIL model can be used to describe a variety of existing peer-to-peer search networks. For example, a supernode network can be represented using FSLs and NILs, as shown in Figure 1a. Supernodes (the central nodes in the figure) are connected using FSLs ($\Longrightarrow$). Non-supernodes ("normal nodes") are connected to supernodes using one FSL and one NIL ($\dashrightarrow$). Then, supernodes have a copy of the indexes of normal nodes, and when a normal node generates a search, it is forwarded to all of the supernodes.

Another example is a network of peers that construct and forward index updates to other peers. Global indexing servers collect and store all of the updates, and peers can search for archived content at the indexing servers. Such a network is shown in Figure 1b. As the figure shows, updates flow along FILs ($\Rrightarrow$) between nodes and to the central index servers. Peers can search the index server, and these searches are carried by NSLs ($\longrightarrow$). The global indexing topology is similar to the Usenet structure, where updates flow around the network and centralized servers (such as DejaNews, now known as Google Groups) retain the updates and an-

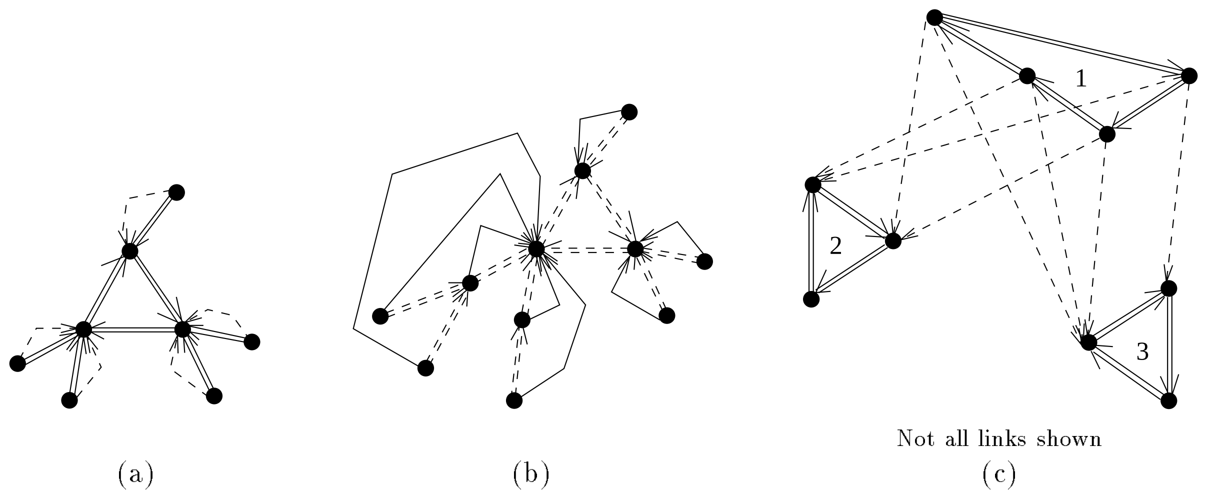


Figure 1: Networks represented using the SIL model: (a) supernodes, (b) Usenet and (c) parallel search clusters

swer queries.

3 Discovering topologies

The SIL model is useful for visualizing existing topologies and network organizations. However, the model's simplicity and generality also allows us to use it to suggest and study new and novel topologies. Using the same link types that form the basic building blocks of existing networks (such as supernodes), we can construct networks that have different and desirable properties.

One novel topology that we have found using SIL is called *parallel search clusters*. An example is shown in Figure 1c. In this architecture, peers are organized into search clusters of FSLs, similar to the Gnutella pure search network. Separate clusters are joined by index links, either FILs or NILs. Figure 1c shows three clusters, with NILs connecting cluster 1 to the other clusters. This graph would also have outgoing NILs from clusters 2 and 3, but we have omitted them for clarity. If the nodes in a cluster collectively have indexes for all the nodes outside the cluster, full coverage can be achieved even though nodes only directly search other nodes in their own cluster.

A parallel search cluster network has several advantages when compared to other network topologies. For example, in a cluster network, the processing required to answer queries is shared among all the nodes in the network. In contrast, in a supernode network, each supernode must process all of the queries in the network, and may become overloaded. If there are few or no nodes with very high capacity, then a supernode network is simply not feasible. Even if some nodes have much higher capacity than others, a parallel cluster network offers more flexibility by allowing nodes to contribute whatever they can. In a supernode network, a node either handles all searches or no searches.

Another advantage of parallel cluster networks is that they can be tuned depending on the load in the network. For example, index updates might occur much more frequently than searches. Then, the network should be constructed with a few large clusters, so that there are relatively fewer index links and thus fewer indexes to be updated. Moreover, if clusters are larger, each node is responsible for fewer indexes, and the load on individual nodes is further reduced. On the other hand, if there are many more searches than updates, the network should be constructed as a large number of small clusters. In this case, each node only has to handle searches from a few nodes and the search load on each node is reduced. Thus, we can change the number of clusters to minimize total load on nodes. Supernode networks are less flexible. If searches comprise the bulk of the load, we cannot increase or decrease the number of supernodes to spread the search load, since every supernode handles all searches regardless of how many supernodes there are.

We have conducted simulation studies to quantify these advantages. In our simulations, we constructed networks that followed the parallel cluster model, and compared them to supernode networks and pure search networks (e.g. Gnutella). The full details of our experiments are outside the scope of this position paper; see [3]. As an example, we evaluated the load on nodes in parallel cluster networks versus supernode networks and pure search networks (such as Gnutella). The maximum load on a node in a cluster network was significantly lower (by up to a factor of 7) than the maximum load on a node in a supernode or pure search network. At the same time, the average load on a node in a cluster network was comparable to the average load on a node in a supernode network. In other words, the high load on supernodes was avoided without significantly overloading the average node in the network. These results illustrate that the SIL model is useful for identifying networks that are more effective in some cases than existing architectures.

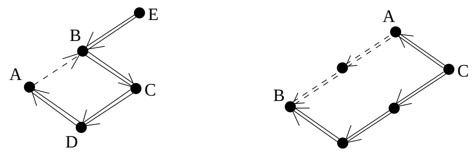
4 Research challenges

The SIL model also provides a framework for examining a variety of interesting research problems in the area of studying and optimizing search networks.

4.1 Topological properties

One research challenge is to identify "desirable" properties of search networks, so that we can discover topologies and build networks that exhibit these properties. We can use SIL as a method for expressing and studying these properties.

Consider, for example, the problem of minimizing the load on peers in the system. One cause of load is *redundancy*, which we define as peers performing work that is duplicated by other peers in the system. A peer A may try to shed load by replicating its index to peer B, so that B can answer queries over A's content without requiring processing on the part of A. However, if A still receives and processes queries that are also being answered by B, then A is doing unnecessary, redundant work. We can specify a general property of SIL graphs as follows:



(a) (b)

Figure 2: Features that cause redundancy: a. one-index-cycle, b. search fork

- *Redundancy* exists in a SIL graph if a link can be removed without reducing query coverage.

In other words, if a link is carrying messages to A, and A need not process these messages in order to ensure that the query is answered, then there is redundancy.

We can make this property more concrete by identifying topological features of SIL graphs that lead to redundancy. One feature that causes redundancy is a specific type of cycle called a *one-index-cycle*: a node A has an index link to another node B, and B has a search path to A. An example is shown in Figure 2a. This construct leads to redundant processing, since B will answer queries over A's index, and yet these queries will be forwarded to A who will also answer them over A's index. More formally, a one-index-cycle fits our definition of *redundancy* because at least one link in the cycle can be removed without affecting coverage: the index link from A to B.

Another feature that causes search/index redundancy is a *search fork*: a node C has a search link to A and a search path to B that does not include A, and there is an index path from A to B. An example is shown in Figure 2b. Again, A will process any searches from C unnecessarily, since B can process the queries for A. The redundant link in this example is the link $C \Longrightarrow A$. We specify that there is a search path from C to B that does not include A because if the only path from C to B included A there would be no link that could be removed without reducing coverage.

Research is needed to identify other graph properties that enhance the efficiency of search networks. Search forks and one-index-cycles can be avoided when constructing supernode networks, and yet supernodes can still become overloaded as the network grows. By analyzing the strengths and weaknesses of various network architectures, we can elucidate desirable or undesirable graph properties, and apply them when constructing networks.

In summary, there is the challenge of defining desirable properties (like redundancy), and the challenge of identifying the topological features (e.g., no one-index cycles and no search forks) that embody those properties. Knowing the properties and features will make it easier to construct "good" SIL networks.

4.2 Dynamic search networks

In the discussion so far, we have assumed that graphs are static. In order to properly model real peer-to-peer networks, we must also represent the process of nodes joining and leaving. Such dynamic networks can easily be modeled by SIL: adding a new node and links to the graph represents a peer joining, while removing a node and all of its incoming and outgoing links represents a node leaving.

However, a dynamic network presents an interesting research question: when a node joins, what connections should it make? The traditional approach is to pick a particular architecture (such as supernodes or parallel search clusters) and force nodes to join in such a way as to preserve that architecture. A different and novel approach is to allow a node to join in any way it likes, creating links as appropriate, as long as it does not introduce undesirable topological features. For example, we might specify that a node could join a network as long as it does not introduce a one-index-cycle or a search fork. If a node could verify that adding a link did not create one of those features, then it could join the network in an ad hoc way while preserving desirable properties such as efficiency or fault tolerance. Nodes could form whatever connections they needed without damaging the efficiency of the network, and the network would evolve and adapt to best meet the needs of member nodes and changing load conditions.

If a dynamic network is not tied to a specific topology, then the topology can more easily change over time depending on the needs of the network. For example, we have begun to study ways for overloaded nodes to shed load by simply disconnecting from some of their neighbors. These neighbors would then reconnect to other, less loaded nodes, spreading the work around the network more evenly. Moreover, a node may find that it is overloaded with search messages, and drop only search links. Then, neighbors would be encouraged to replace those search links with index links. In this way, the nature of the network would change from search-centric to index-centric as a reflection of the current load conditions.

SIL provides a framework for trying out different ad hoc graph-construction methods and examining their effects. We have examined several techniques, and our results are described in more detail in [2]. Our experiments indicate that ad hoc methods can be effective in many situations in significantly improving the efficiency of the network.

5 Related work

Optimization of peer-to-peer search networks is a hot topic among distributed systems researchers. Some investigators are examining techniques for more effective searching of existing networks, for example using random walk searches [7]. Others have examined how to build better versions of existing networks [12, 9], how to "fix-up" an inefficient network [8], or how to better index an existing network [4]. The SIL model is an attempt to provide a framework for generalizing these approaches. Complementary to the SIL model is work that explicitly models the location of content, such as systems that attempt to replicate content in the most effective way [1]. It may be useful to extend our model to model content as well as nodes and links.

Another current research trend is to focus on networks that allow users to locate objects by name instead instead of by content. Such networks are typically designed as distributed hash tables (DHT) and several DHT architectures have been proposed [11, 10]. There are still interesting research questions in the space of flooding-based networks, and as discussed in Section 1, the advantages of DHTs do not mean that we should not study SIL. Yet another focus of many investigators is designing networks for privacy, security or anonymity (such as SOS [6] or FreeHaven [5]). Our focus on content discovery is complementary to these approaches.

6 Conclusion

The SIL model is a general mechanism for describing the topology and properties of peer-to-peer search networks. It is useful as:

- A framework for evaluating networks in terms of metrics such as efficiency or fault tolerance.
- A tool for discovering new and interesting network architectures.
- A mechanism for defining and studying desirable topological properties of networks.
- A platform for studying new ways of constructing and maintaining dynamic networks.

References

[1] E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks. In *Proc. SIGCOMM*, August 2002.

[2] B.F. Cooper and H. Garcia-Molina. Ad hoc, self-supervising peer-to-peer search networks, 2003. Technical Report.

[3] B.F. Cooper and H. Garcia-Molina. Sil: Modeling and measuring scalable peer-to-peer search networks, 2003. Technical Report.

[4] A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer systems. In *Proc. Int'l Conf. on Distributed Computing Systems (ICDCS)*, July 2002.

[5] R. Dingledine, M.J. Freedman, and D. Molnar. The FreeHaven Project: Distributed anonymous storage service. In *Proc. of the Workshop on Design Issues in Anonymity and Unobservability*, July 2000.

[6] A. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure overlay services. In *Proc. SIGCOMM*, Aug. 2002.

[7] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *Proc. of ACM International Conference on Supercomputing (ICS'02)*, June 2002.

[8] Q. Lv, S. Ratnasamy, and S. Shenker. Can heterogeneity make gnutella scalable? In *Proc. of the 1st Int'l Workshop on Peer to Peer Systems (IPTPS)*, March 2002.

[9] G. Pandurangan, P. Raghavan, and E. Upfal. Building low-diameter P2P networks. In *Proc. IEEE Symposium on Foundations of Computer Science*, 2001.

[10] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proc. SIGCOMM*, Aug. 2001.

[11] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. SIGCOMM*, Aug. 2001.

[12] B. Yang and H. Garcia-Molina. Designing a super-peer network. In *Proc. ICDE*, March 2003.