SETS : Search Enhanced by Topic Segmentation

Mayank Bawa
Stanford University
bawa@db.stanford.edu

Gurmeet Singh Manku
Stanford University
manku@stanford.edu

Prabhakar Raghavan
Verity Inc.
pragh@verity.com

ABSTRACT

We present SETS, an architecture for efficient *search* in peer-to-peer networks, building upon ideas drawn from machine learning and social network theory. The key idea is to arrange participating sites in a *topic-segmented* overlay topology in which most connections are *short-distance*, connecting pairs of sites with similar content. Topically focused sets of sites are then joined together into a single network by *long-distance* links. Queries are matched and routed to only the topically closest regions. We discuss a variety of design issues and tradeoffs that an implementor of SETS would face. We show that SETS is efficient in network traffic and query processing load.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Clustering, Search process, Selection process; H.3.4 [**Systems and Software**]: Distributed Systems, Performance Evaluation (efficiency and effectiveness)

General Terms

Algorithms, Performance, Design

Keywords

Peer-to-Peer (P2P), distributed information retrieval, small world networks, topic-driven query routing, topic segments.

1. OVERVIEW

Peer-to-peer (P2P) networks have received considerable attention recently. Such networks are characterized by a very large number of participating sites that span wide area networks and cooperatively share content with each other. The first generation of P2P networks focused on collections of music files. However, the real potential for such networks lies in the sharing of valuable enterprise documents.

As a first step towards building P2P applications, researchers have proposed protocols for performing efficient key *lookups* by constructing Distributed Hash Tables (DHTs) [25, 30]. A DHT is simply a hash table that is partitioned among a collection of sites. A lookup for a key is routed efficiently to that site which is responsible for storing that key. However, a *search* query is more complex than simple key lookups. It is not clear how a key *lookup* service could be used to build an efficient *search* application.

Distributed Information Retrieval [4] has studied problems associated with searching distributed heterogeneous repositories. Researchers have focused on identifying architectures [2, 10, 15] and models [5, 28, 37] that can search hundreds of repositories. In this context, researchers have investigated common resource description languages [29], schemes for repository selection [12] and merger of ranked lists [36]. Heterogeneity in search interfaces has led implementors to converge on *mediator-based* architectures [1, 3, 23] in which wrappers for each participant repository rewrite the query and interpret the ranked lists returned.

P2P networks are distributed. However, their characteristics are markedly different from Distributed Information Retrieval systems: (a) The number of participants (usually PCs) is in tens to hundreds of thousands, (b) Each site downloads a common binary. This allows an architect to enforce a common resource description language and search interface, (c) P2P networks are dynamic. Site lifetimes are short (a few hours) and arbitrary (whims of the site owner), and (d) P2P networks are low-cost systems built on the principle of utilizing unused computing resources. Often, there is no central server that acts as a mediator for statistics collation, repository selection, query rewriting and ranked list merging. A common strategy adopted by popular P2P networks like Gnutella, KaZaa and Morpheus is to retrieve results by flooding a query to as many participants as possible. This clearly wastes network resources.

In this paper, we bridge the gap between Distributed Information Retrieval and P2P networks by proposing SETS, a topic-segmentation based network that provides efficient *search* on P2P networks. The architecture is modular and allows advances in distributed information retrieval to be "plugged" in cleanly. We provide a rigorous evaluation of SETS. In the course of such analysis, we are able to assign a measure of the quality of document clustering that is simple, precise and exact in our context.

1.1 Topic-Segmented Networks

The philosophy underlying SETS is to arrange sites in a network such that a search query probes only a small subset of sites where most of the matching documents reside. In particular, SETS partitions sites into *topic segments* such

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGIR 2003, July 28 — August 1, Toronto, Canada

Copyright 2003 ACM 0-12345-67-8/90/01 ...$5.00.

that sites with similar documents belong to the same segment. Each topic segment has a succinct description called the *topic centroid*. Sites are arranged in a segmented network that consists of two kinds of links. Short distance links connect sites within a segment. Long distance links connect pairs of sites from different segments. When a search query is initiated, it is forwarded to other sites using a *topic-driven routing protocol*. First, topic centroids are used to select a small set of relevant topic segments. Next, the selected segments are probed in sequence. A probe to a particular segment proceeds in two steps: First, the query is routed along long distance links to reach a random site belonging to the target segment. Next, the short distance links are used to propagate the query to all/most/few sites within a segment.

Can the above philosophy enable efficient search over P2P networks? The answer depends on (a) the quality of the topic segments and (b) the efficiency of topic-driven routing. The two goals can be met only if we architect each of the following building blocks efficiently:

- A *Topic Segment Construction*: How do we partition sites into topic segments?
- B *Topic Segment Maintenance*: How are segments maintained as sites join and leave, and/or their document collections change over time?
- C *Topic Segment Selection*: Which segments (and in what sequence) do we probe to answer a query?
- D *Global Routing*: How is a query forwarded to (some member of) a particular segment?
- E *Local Routing*: Once a query reaches some site within a specific segment, how is it propagated to all/most/few sites within that segment?

The first three building blocks relate to topic segments and the last two to topic-driven query routing. Each building block poses interesting problems that we discuss next.

A. Topic Segment Construction

We need to group sites with similar content together into topic segments. *Heterogeneity* and *autonomy* of sites pose challenges to good topic segmentation. Sites often have documents drawn from diverse topics. Autonomy of sites implies that the system cannot dictate the placement of documents. A good segmentation should yield high recall from low *query processing cost*, i.e., the average number of sites that are probed to evaluate a query.

Note that topic segmentation poses a complex clustering problem with a surprisingly clean objective function — *query processing cost*. We emphasize that the objective function is simple and *exact*. Typical studies of clustering formulate it as an optimization work with objective functions that require a leap of faith for interpretation in the application domain. Our measure remains cleanly related to our application even though our methodology envelops issues like connections and routing that go beyond clustering.

B. Topic Segment Maintenance

In a P2P network sites leave and join without notice. Moreover, the set of documents at each site changes over time. Maintaining consistent global knowledge of topic centroids in the face of such changes is challenging.

Two issues that arise due to centroid re-computation are that (a) new centroids have to be disseminated to all the sites, and (b) sites re-assign themselves to new segments if they discover that they no longer belong to their current segment. The first problem requires an efficient mechanism for informing all sites about new centroids. The second problem requires a solution that ensures that site re-assignment is done smoothly over time. For stability, it should not so happen that a large set of sites suddenly decide to migrate.

C. Topic Segment Selection

When a query is initiated, we need to identify a small set of topic segments that will be probed. The challenge lies in designing an algorithm for ranking segments by matching a given query against their descriptions. Recall depends crucially on the choice of segments identified for probing.

Once a small set of segments are identified, the query is forwarded to each segment in the set in two steps: Global Routing forwards the query to some site in the target segment. Then, Local Routing propagates the query to a subset of sites within the segment.

D. Global Routing

The goal of Global Routing is to forward a query to a target topic segment using long distance links. The challenge lies in designing a scheme that maintains network connectivity and routes queries in few hops without requiring too many long distance links per site. Moreover, the scheme should work in the presence of frequent arrival and departure of sites without notice.

E. Local Routing

Once a query reaches some site within a segment, it is propagated to a carefully chosen subset of sites within that segment using short distance links. The design of intra-segment routing protocols is an interesting problem by itself. The protocol depends upon the particular domain of documents and the expectations from the search system on the whole.

Layout of the Paper: Section 2 explores related work. Section 3 describes the SETS architecture. Section 4 discusses the experimental methodology. Section 5 contains experimental results. Section 6 concludes the paper.

2. RELATED WORK

Gnutella and Freenet are the best-known examples of P2P search networks. SETS retains many of the best aspects of these systems (decentralized operation, sites operating as true peers and fully dynamic sites). Gnutella relies on broadcasts to answer queries. Since broadcast is unscalable, Gnutella adopts heuristics (e.g., time-to-live fields) that constrain query propagation radius and result in reduced recall. Freenet replicates documents across the network in the hope that documents matching a query can be located close to the querying site. Other improvements have been suggested [7, 9, 22, 38]. However, there are no guarantees on recall.

Distributed Hash Tables (DHTs) [18, 25, 30] arrange the network to answer lookup queries. pSearch [31] proposes using DHTs for search. A global index for each term is placed at the site that is responsible for the term. It remains to be seen if network traffic for index updates necessitated by site joins, leaves and document movements can be supported.

The preferred architecture for Distributed Information Retrieval has been mediator-based. A mediator collects metadata from different participating collections. Queries are sent to the mediator who selects relevant collections, rewrites the query for each collection and forwards the rewritten query to the selected collections [1, 3, 23]. This architecture was shown to scale to tens of thousands of collections with best performance under a domain-influenced "logical" organization [8]. Correspondingly, sites have also been clustered around "global concepts" [20, 24] and clusters arranged in a hierarchy [27, 35] to allow retrieval through navigation or search. Information about each cluster is collected at a mediating router which then routes queries to relevant clusters. P2P networks tend to be characterized by the absence of such stable infrastructural units.

The principle of topic segmentation in SETS is based on the *cluster hypothesis* which states that "closely associated documents tend to be relevant to the same requests" [32]. Our measure of efficacy is analogous to *cluster retrieval* [26] but goes beyond it to concretely quantify and explore the trade-offs between cost and quality entailed.

Xu et. al. [37] demonstrated that organizing collections into topics when combined with language models improves retrieval. As in [26], the assumption is that collections at a single site are large enough that organizing them into clusters is meaningful. We observe that individual collections in P2P networks are small. Documents are already partitioned into sites and cannot be migrated from one site to be placed at another.

Social network theory hypothesizes that information flows occur along ties in a networked society. The theory distinguishes between "strong ties" between close associates and "weak ties" between acquaintances. While strong ties are "short distance" as they stay within groups [11], weak ties are "long distance" connecting people with different social characteristics [14]. In SETS, a site maintains both short and long distance links to discover information.

A social network typically exhibits the *small-world phenomenon* [19]. Pairs of individuals are connected through short chains of acquaintances. Moreover, individuals can often discover such chains. Kleinberg [16] proposed a network topology which allows messages to be routed between arbitrary pairs of sites in few hops. In SETS, topic segments are organized into a similar network.

3. SETS: ARCHITECTURE

A distinguished site A in a P2P network is responsible for specific administrative tasks while it participates in the network as a peer. For example, it pushes out and maintains code-bases, provide an inlet to the network by supplying addresses of a small set of currently active peers, etc. Note that A runs in the background providing *passive* support and is not involved in *active* run-time operations.

3.1 Topic Segment Construction

Deducing a compact representation of documents at a site is an interesting problem by itself. A robust solution is the *unigram model* [4] wherein the representation is either a list of words with their frequencies of occurrence [5, 15] or a statistic derived from them [34]. Better results were obtained recently by using a set of representative related words [13] or using language modeling with collections organized by clustering documents [37]. However, we use the simpler unigram model of representation as follows.

Each document is represented as a term vector normalized to unit length. The terms of a document are the stemmed words that occur within it. Stop words and highly frequent words are removed from the term vector. Each remaining term is assigned a value in the vector equal to its *weighted term frequency* [26] which is $1+\log tf$. This statistic has two advantages as compared to the tf-idf measure: (a) it produces higher quality clusters [26], and (b) it does not require global information for computation. A site has a collection of documents and is represented by a term vector of unit length called the *site vector*. It is formed by normalizing the sum of all term vectors for documents at that site.

We experimented with two ideas for generating topic segments: (a) cluster document vectors, and (b) cluster site vectors. In either case A generates C clusters where C is a parameter fixed at the outset. Each cluster corresponds to a topic segment and their centers constitute topic centroids. Knowledge of the C topic centroids is global.

When a new site joins the network, it obtains the current set of C topic centroids from A. It then computes its own site vector and identifies the segment whose centroid is closest to its site vector. The site then uses Global Routing (Section 3.4) to obtain the identity of some site that currently belongs to this segment. Short distance links are established within this segment as described in Section 3.5. Long distance links are established as described in Section 3.4.

3.2 Topic Segment Maintenance

The set of C topic centroids changes over time as sites join/leave and their document collections change. We now discuss how centroids are recomputed and disseminated.

For dynamic maintenance of centroids, two pieces of information are exchanged between A and all other sites: (a) Every site sends its initial site vector (and changes over time to its site vector) to A and (b) A hands the new topic centroids to all sites. Clearly, recomputing and disseminating topic centroids for every change in topology or document collection at a site is impractical. If each site communicated each small update to A individually, A would easily be swamped. Also, A would require very high bandwidth to disseminate recomputed clusters for each change. We propose a simple and clean solution for these problems: *leases.*

Leases

Topic centroids are recomputed at regular intervals of time at A. The interval between successive re-computations is a tunable parameter T. Whenever the current set of topic centroids is made known to a site, the set is tagged with a *lease*: a random number drawn uniformly from the interval $[1, T]$. A site contacts A only when its lease expires. At that time, it informs A about changes in its site vector, and A hands it the current set of centroids, along with a new lease.

What do leases yield? A site now contacts A roughly twice every T time units rather than for every small update. Since leases are drawn uniformly from $[1, T]$, the times at which sites contact A are *staggered*. The bandwidth requirements at A are thus spread over the entire lease period. At any moment, sites belong to at most two different sets of topic centroids (corresponding to successive incarnations of centroids). From the perspective of global consistency, there is a problem: sites that remember the old set of centroids might no longer be assigned to the closest topic segment (among the new set of centroids). This may result in increased *query processing load* as the number of sites probed per query is likely to increase because of an imperfect view of the clustering. There is also a loss in recall because the cluster to which a site is currently (incorrectly) assigned might not be probed at all for queries that match documents at this site. Provided T is chosen such that successive sets of centroids do not differ significantly, the increase in query processing load and loss in recall is not significant.

Migration to New Segment

When a site receives a new set of centroids, it might realize that it no longer belongs to the topic segment that it currently lies in. The site then deletes itself from the network and re-joins. Deletions and joins are carried out as per Global Routing and Local Routing protocols (Sections 3.4 and 3.5). We assume that T was chosen such that topic centroids do not move significantly. This means that over T time units, only a small fraction of sites would have to migrate to new topic segments. The use of leases makes the load on the network due to such migration fairly uniform over the network lifetime. It is very unlikely that a large collection of sites suddenly decides to migrate together.

3.3 Topic Segment Selection

A query is a set of terms and can be represented by a normalized term vector called the *query vector*. When a query is issued, the similarity between the query vector and each topic centroid is computed. The similarity scores are used to rank the C segments in descending order with ties broken arbitrarily. The top $R \leq C$ segments are then deemed relevant, where R is a tunable parameter.

The similarity function used has immense bearing on recall. The CORI algorithm [5, 12] has been shown to perform well, but requires global information and high maintenance traffic when documents have been clustered [33]. We experimented with the simpler cosine distance function which needs minimal information for computation.

A query message consisting of the query vector and the target topic segment id is composed, one for each of the R relevant segments. The R query messages are issued in parallel. Each is first routed to some site in the target segment using Global Routing. Next, Local Routing further propagates the query within a segment. We experimentally show (Section 5) that a small value of R suffices for high recall.

3.4 Global Routing

Given a collection of sites, the goal of Global Routing is to forward a query message along *long distance links* to some site that belongs to the target segment. In a companion paper, we describe a randomized protocol [18] for Global Routing. The key idea is to arrange the sites in a unit ring, ordered by segment id. Each site has a link to each of its immediate neighbors and a small number of long distance links drawn from a family of harmonic distributions. We show that if each site has k links, then a query message can be routed in $O((\log^2 n)/k)$ hops, where n is the total number of sites. The protocol is simple yet scalable that provides low latency even when sites join and leave frequently. Global Routing is further investigated in [17].

3.5 Local Routing

Once a query reaches some site within a target segment, Local Routing propagates the query further within the segment. We argue that Local Routing is heavily influenced by the specific domain in which SETS is deployed.

Different domains have different expectations and impose different constraints on the search systems employed. For example, when sharing music files, users are satisfied with tens of results. In contrast, a network for sharing patent information requires close to 100% recall. Similarly, when a network is formed on PCs with $56Kbps$ modems, users require search to be efficient in its bandwidth usage. SETS implementors would devise Local Routing tuned to their domain. We believe that the rest of SETS is flexible enough to impose very few restrictions on the design of Local Routing.

We now describe one possible architecture for Local Routing. The idea is to arrange sites within a topic segment as a random graph with constant degree m, where m is a parameter fixed at the outset. Our evaluations (Section 5) were performed assuming such a Local Routing architecture:

- *Site Insertion*: A new site establishes short distance links with m sites chosen uniformly at random from among the current members of the segment, where m is a parameter fixed at the outset. With high probability, a random site to connect to can be discovered by carrying out a random walk of size $O(\log n)$ over short distance links [21], where n is the current number of sites in the segment.
- *Site Deletion*: A departing site simply terminates its short distance links. Its former neighbors establish links with other members of the segment.
- *Query Propagation*: Queries are forwarded to all members of a segment by flooding: each site that receives a query first checks if it has seen this query before. If so, the query is dropped. Otherwise, the query is forwarded to each of its neighbors except the one from which the query was just received.
- *Query Evaluation*: Upon receiving a query, a site evaluates it against its documents. Matches are directly reported to the site where the query originated. Failure to produce any matches is not reported to any site.

4. EVALUATION METHODOLOGY

SETS has two major components: topic segmentation and topic-driven query routing. In this section, we discuss how the quality of each is evaluated.

Our evaluation differs from previous research in distributed retrieval in several ways. First, we utilize testbeds with tens of thousands of sites (albeit with fewer documents per site) for our experiments. Second, whereas other efforts have studied the effect of organizing documents in the testbed corpus by source, publication date, etc. [12, 33, 34], we study the impact of having documents organized into sites according to the humans who created them. Finally, we quantify and explore aspects of cost vs quality tradeoff in vector-space retrieval when less than 100% recall is acceptable.

Performance Metrics

Our focus is not on providing better merging functions to improve precision, but rather on topic segmentation and its effect on search efficiency for any given level of recall. Given this focus on the efficacy of topic segmentation, we restrict ourselves to studying recall. Specifically, we measure recall as a function of network load under a simple (binary) notion of whether a document matches a query.

The quality of topic segmentation is measured by *query processing cost*, defined as the average number of sites that are probed to evaluate a query. If the quality of topic-segmentation is poor, irrelevant sites will evaluate the query. Thus more topic segments are explored for a given level of recall. A good segmentation, on the other hand, would assign relevant sites to the same segment yielding the same level of recall from exploring a few segments.

The quality of topic-driven query routing is measured by *bandwidth* and *latency* per query. Bandwidth requirements are proportional to the total number of messages sent. Latency is the time elapsed from query issue to the time at which the first answer is received.

Document Sets

We evaluated SETS over three different datasets:

A *TREC-1,2-AP*: Documents from AP Newswire in TREC CDs 1 and 2 include text and author fields. We deemed each author to be a site and associated documents to sites in the natural manner. Documents that did not have a valid author field were excluded. The text of an article was used to construct its document vector. This resulted in 79,180 documents shared by 1,834 sites.

B *Reuters*: The news articles that comprise the Reuters Corpus (Vol. 1) include text and author fields. We deemed each author to be a site and associated documents to sites as above. The text of an article was used to construct its document vector. This led to 109,500 documents shared by 2,368 sites.

C *CiteSeer*: Since the above datasets are not large enough to evaluate SETS at the truly large scales it is capable of, we compiled a new dataset as follows. We crawled the Computer Science Directory of the CiteSeer Library [6] to obtain a list of research papers. For each paper, we recorded its title, abstract and URL from which Citeseer obtained it. We deemed each unique URL to denote a site. The set of papers available off a URL is the collection of documents for the corresponding site. This resulted in 478,256 documents shared by 83,946 sites.

Query Generation and Answer Determination

The queries for *TREC-1,2-AP* were obtained from TREC-3 ad hoc topics (151-200). The text in the title fields was stemmed and stop-words were removed to obtain a conjunction query. This query set thus had 50 queries with average length 3.56 and standard deviation 1.03. The answer for a query comprised of documents in the dataset that were determined to be relevant for the corresponding topic by TREC-3 ad hoc query assessors. We note that topics 151-200 were chosen because relevance judgments for each of these included documents in *TREC-1,2-AP*.

Each query for *Reuters* and *CiteSeer* was obtained by first choosing a document at random and then choosing 3 terms uniformly at random from its vector. Our query set comprised of 100 such queries. The answer for a query comprised of all documents that contained *every* term in the query.

5. PERFORMANCE EVALUATION

All tests were run on all the datasets; where the results are similar we present results on one of the datasets for brevity.

5.1 Topic Segment Construction

Our measure of quality of topic segmentation is *query processing cost*. We note that this cost is tied to the Segment Selection and Local Routing scheme used. Given a query, the Segment Selection first ranks the centroids by similarity to the query and then chooses a small set of segments to probe. It is Local Routing that determines which sites within a segment will actually be probed. Here we assume that when the Segment Selection decides to probe a segment, Local Routing will probe *all sites* within that segment.

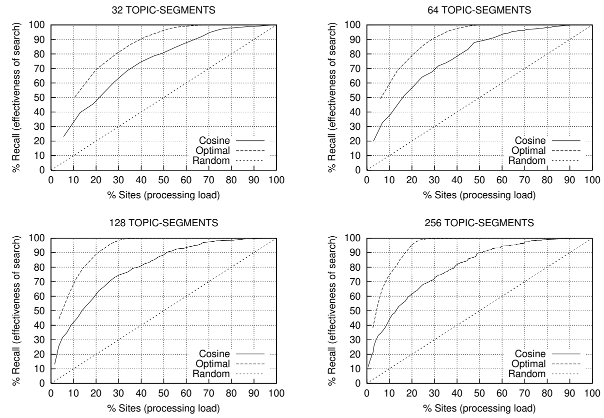

Figure 1: Topic-segmentation on *TREC-1,2-AP* obtains higher recall by processing queries at fewer but relevant sites.

Recall vs Query Processing Cost

We studied three Segment Selection schemes: a) RANDOM: topic segments were ordered randomly, b) COSINE: segments were ordered by the cosine similarity between the query vector and topic centroids and c) OPTIMAL: For each query, segments were ordered by the ratio (number of matching documents) : (number of sites in segment). The RANDOM ordering serves as a baseline to compare gains against a system (e.g., Gnutella) that does not use topic segmentation. The OPTIMAL ordering on the other hand is derived from an omniscient central server with complete knowledge of all documents and segments and is further allowed to pick the best ordering for each query at run-time. Clearly such ordering is infeasible in practice, but serves as a useful benchmark for comparisons as a theoretical limit.

Figure 1 plots recall vs query processing cost. The topic segments used in the experiment were constructed by clustering site vectors of *TREC-1,2-AP*. The curves are drawn for various choices of the number of topic segments C ranging from 32 to 256. We observe that COSINE outperforms RANDOM substantially, returning $65-75\%$ recall by exploring *only* 30% of the network. We discuss the OPTIMAL ordering curves in Section 5.3.

Basis of Clustering: Documents versus Sites

In Figure 2, we plot recall vs query processing cost for *CiteSeer* with C ranging from 64 to 512. The curves are drawn for two methods for topic segment construction: (a) Clustering of document vectors, and (b) Clustering of site vectors. The OPTIMAL, SITE-BASED COSINE and RANDOM curves are drawn for corresponding Segment Selection schemes on a network clustered by sites. The DOCUMENT-BASED COSINE curve is drawn for COSINE Segment Selection on a network clustered by documents. We see that site vector clustering outperforms document vector clustering across a broad range of C values — it consistently returns higher recall for the same processing cost.

Why does site vector clustering outperform document vector clustering? The reason is that the documents at a site are

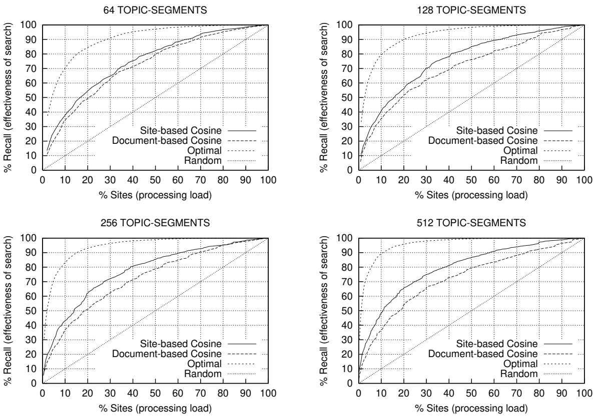

Figure 2: Topic-segmentation on *CiteSeer* repository shows that site-based clustering outperforms document-based clustering.

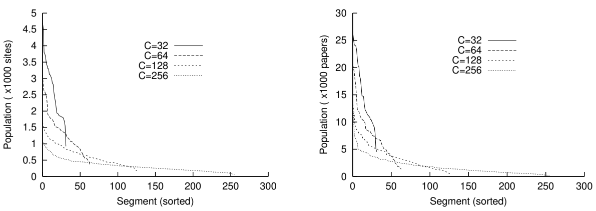

Figure 3: *Left:* Skew in segment site population. *Right:* Skew in segment document population. Both are for site-based clustering of *CiteSeer* repository.

often drawn from diverse topics. If clusters are derived from document vectors, a site tends to get assigned to that topic that dominates its document collection. Since site classification is influenced greatly by the dominant topic, queries for documents corresponding to subordinate topics suffer. Site vector clustering alleviates the situation by accommodating this heterogeneity. Clusters that are formed now can reflect a mix of topics and sites are assigned according to their aggregate collections. Documents at such sites can be easily located leading to improved performance.

We also observe that the site-based curves are markedly similar to those obtained on *TREC-1,2-AP* in Figure 1. Henceforth, we report only on results obtained on the larger scale *CiteSeer* noting that similar results were obtained on *TREC-1,2-AP* and *Reuters* datasets.

Quality of Clusters

In Figure 3, we plot the distribution of segment populations for *CiteSeer* using site-based clustering. The number of topic segments C is varied from 32 to 256. We observe that for small values of C the distribution of both site and paper populations is skewed. However, as C increases the skew decreases and segments tend to become equi-sized.

In Table 4, we depict the 5 most significant terms in a sample of 4 topic segment for C=32 with site-based clustering. The labels indicated in parentheses were manually assigned. As can be observed, site-based clustering is quite successful in forming focussed segments.

1 (Compilers)	2 (Crypto)	3 (Databases)	4 (Internet)
compil	secur	queri	java
garbag	attack	databas	web
polymorph	authent	retriev	server
haskell	cryptograph	xml	internet
loop	encrypt	schema	client

Figure 4: Sample topic segments obtained by clustering site vectors for the *CiteSeer* repository.

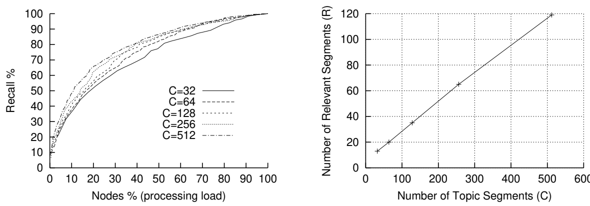

Figure 5: *Left:* Recall curves pull up for increasing C, however with diminishing returns. *Right:* Values of R for 70% recall increase almost linearly with C.

Choice of Number of Clusters, C

In the left sub-figure of Figure 5, we plot recall vs query processing cost for *CiteSeer* using site-based clustering and COSINE Segment Selection. The number of topic segments C is varied from 32 to 512. We observe that recall increases as C increases but with diminishing returns. As C increases, topic segments become smaller and more focussed resulting in lowered cost of probing a segment. However, such gains come at the cost of increased processing load at administrator site A. Thus fixing C involves a trade-off between increased recall and increased resource consumption at A.

5.2 Topic Segment Maintenance

SETS requires an administrative site A that provides passive support for maintaining a topic-segmented network. Let us study the communication and processing demands that SETS imposes on A. Suppose there are n sites participating in the network. Each of the n sites will send changes in their site vector to A and accept C new topic centroids over a duration of T minutes. The administrator A has to cluster site vectors into C clusters during this time.

Assume an average of t terms per site, $4B$ term ids, and *no* compression of term vectors. As described in Section 3.2, on average each site contacts A twice in T minutes. On average, A has an in-bound traffic of $2 \times 4t \times n = 8tn$ bytes in T minutes. Similarly, assuming an average of t' terms per topic segment description, A has an out-bound traffic of $2 \times 4t' \times C = 8t'C$ bytes in T minutes.

Let us assume that $t = t' = 1,000$ terms, $T = 15$ minutes, $C = 256$ and $1.54M$ bits per second T1 line. Assuming that 20% of the bandwidth is consumed by networking overhead (TCP/IP headers), A can support $n = \frac{80\% \times 1.52 Mbps \times T mins}{8 \times 8(t+t')} - C = 8,827$ sites. For a corporate setting with a $44.736M$ bits per second T3 line, A can support $n = 256,418$ sites. We note that these numbers are in fact pessimistic as they do not account for any optimizations. For example, most site vectors will *not* change in successive $T = 15$ minutes. Such sites can save bandwidth by sending a "no change" message to A. The bandwidth requirements for A are thus reasonable.

Let us now consider processing costs at A. Suppose A uses *k-means* algorithm to cluster site vectors into C clusters. Folklore has it that k-means converges in 5-6 iterations. We implemented a cache-aware disk-resident k-means algorithm that produces C=32 clusters in about 5 minutes, and C=512 clusters in about 15 minutes for $83,946$ sites on a Pentium II processor with 256MB memory. Fresh clusters can thus be generated within the time-scales involved.

5.3 Topic Segment Selection

When a query is generated, Segment Selection selects a small subset of topic segments that appear most promising. Ranking of topic segments is done by comparing the query vector with topic centroids. The segments could be probed in sequence or in parallel.

Impact of Ranking Function

We have already mentioned the three methods studied for ordering segments for probing. Figure 1 shows that the performance of the OPTIMAL is extremely good, with recalls between 90% and 99% being obtained from evaluating the query at just 30% of the network. The gap in the OPTIMAL and COSINE curves indicates the difference between the distributed implementation in SETS and an omniscient central ordering that can compute the best ordering for each query and topic segmentation. There is no guarantee that a distributed system like SETS can attain the performance of OPTIMAL, but we do note that no approach based on topic segmentation can exceed the performance of OPTIMAL.

Choice of R, the Number of Relevant Segments

Once an ordering of cluster segments has been determined, queries are issued in parallel to explore the most relevant R topic-segments. Users are often satisfied with the top few answers. The implementor then has the choice of stopping the query after exploring just the initial R topic-segments. The remaining topic-segments can be explored if the user really would like to obtain more results. Thus, the choice of R depends on the expectations of users in the domain.

What is a good value for R, the number of relevant segments? Consider a network over *CiteSeer* with site-based clustering and COSINE Segment Selection. Let us fix recall at 70%. We simulated SETS for different values of C and computed the number of topic-segments R that were necessary to obtain 70% recall. The results are shown in the right sub-figure of Figure 5. We observe that R increases almost linearly with C over a large range of values for C. We also note that R is roughly a quarter the value of C for $C \geq 200$. This observation suggests a useful rule-of-thumb to us: we can set $R \equiv C/4$ in this design.

5.4 Global Routing

Figure 6 plots the bandwidth and latency used by Global Routing on *CiteSeer* repository. We simulated Global Routing with 4 long distance links per site and a random ordering of topic segments along the ring. We assumed that queries originate from any site at random. We let Segment Selection identify the top ten topic segments for each query (R=10). The union of these segments gives a *query profile*, i.e., a distribution over topic segments. The average latency per query is 8 hops. Latency increases with C, the number of clusters. This is because average cluster size diminishes with increasing C. As a consequence, Global Routing has to home in on a smaller subset of sites, requiring more hops. The average bandwidth shows high variance. This is an artifact of the routing protocol and is discussed in [18].

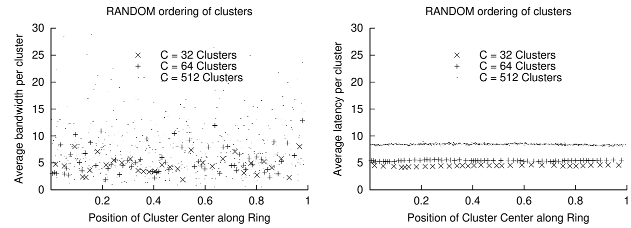

Figure 6: Bandwidth and latency requirements for Global Routing on $83,946$ sites in *CiteSeer* repository.

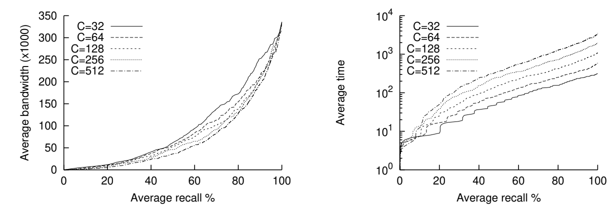

Figure 7: Bandwidth and latency requirements for Local Routing on $83,946$ sites in *CiteSeer* repository.

5.5 Local Routing

We argued earlier that the design of Local Routing is influenced by the particular domain in which SETS is deployed. Yet, its design is of importance as the bandwidth and latency observed here dominate the total cost as we show below.

We simulated Local Routing using the scheme detailed in Section 3.5 with m=4 neighbors per site. For each query in the query set, we let Segment Selection determine the relevance of topic segments and then probed the segments in *sequence*. Figure 7 plots bandwidth and latency vs observed recall on *CiteSeer* with $83,946$ sites.

Bandwidth for a given recall decreases with increasing C. As C increases, topic segments become smaller and more focussed. Fewer segments have to be explored leading to lower bandwidth usage. Also notice that bandwidth costs here are substantially higher than Global Routing. Latency increases with increasing C. The reason is that topic segments are probed in *sequence*. As C increases, there are more topic segments that need to be probed to provide the same recall. The plot also indicates that response times are low as the first answers are received quickly.

Notice also that 70% of recall is attained at a quarter of the total bandwidth and latency costs. Thus, if we altered our scheme to probe just R=$C/4$ top segments, the bandwidth and latency costs would be correspondingly smaller. Hence we conjecture that domain requirements will heavily influence the optimal design for Local Routing.

6. CONCLUSIONS AND FUTURE WORK

We presented SETS, an architecture for efficient search in P2P networks. The underlying philosophy is to arrange participating sites into an overlay network such that queries can quickly reach small regions of the network where most of the matching documents reside. Towards this end, SETS builds a topic-segmented network and employs a topic-driven query routing protocol. We discussed a variety of design issues and trade-offs that an implementor of SETS would face. Through a series of systematic experiments, we showed that SETS provides good recall with good network (small latency and bandwidth per query) and query processing performance. These results clearly suggest that SETS is a viable architecture for organizing content sharing P2P networks.

As future work, we plan to adapt SETS to handle large swings in population characteristics, and develop protocols to change the number of topic-segments at run-time.

7. REFERENCES

[1] D. Barbara and C. Clifton. Information brokers: Sharing knowledge in a heterogeneous distributed system. In *Proc. 4th Conf. on Database and Expert Systems Applications (DEXA)*, pages 80–91, 1993.

[2] C. M. Bowman, P. B. Danzig, D. Hardy, U. Manber, M. F. Schwartz, and D. P. Wessels. Harvest: A scalable, customizable discovery and access system. *Computer Networks and ISDN Systems*, 28(1–2):119–125, 1995.

[3] B. Cahoon and K. S. McKinley. Performance evaluation of a distributed architecture for information retrieval. In *Proc. 19th ACM Conf. on Inform. Retrieval (SIGIR)*, pages 110–118, 1996.

[4] J. Callan. Distributed information retrieval. *Advances in Information Retrieval*, pages 127–150, 2000.

[5] J. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *Proc. 18th ACM Conf. on Inform. Retrieval (SIGIR)*, pages 21–28, 1995.

[6] Citeseer: Scientific literature digital library (http://citeseer.nj.nec.com/cs).

[7] E. Cohen, H. Kaplan, and A. Fiat. Associative search in peer-to-peer networks: Harnessing latent semantics. In *Proc. IEEE Infocom*, 2003.

[8] J. G. Conrad, X. S. Guo, P. Jackson, and M. Meziou. Database selection using actual physical and acquired logical collection resources in a massive domain-specific operational environment. In *Proc. 28th Conf. on Very Large Data Bases (VLDB)*, pages 71–82, 2002.

[9] A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer networks. In *Proc. Intl. Conf. on Distributed Computing Systems (ICDCS)*, pages 23–34, 2002.

[10] P. B. Danzig, J. S. Ahn, J. Noll, and K. Obraczka. Distributed indexing: A technique for scalable, distributed information retrieval systems. In *Proc. 14th ACM Conf. on Information Retrieval (SIGIR)*, pages 220–229, 1991.

[11] S. Feld. Social structural determinants of similarity among associates. In *American Sociological Review (47)*, 1982.

[12] J. French, A. Powell, J. Callan, C. Viles, T. Emmitt, K. Prey, and Y. Mou. Comparing the performance of database selection algorithms. In *Proc. 22nd ACM Conf. on Information Retrieval (SIGIR)*, pages 238–245, 1999.

[13] S. Gauch and J. Wang. A corpus analysis approach for automatic query expansion. In *Proc. 6th Conf. on Information and Knowledge Management (CIKM)*, pages 278–284, 1997.

[14] M. S. Granovetter. The strength of weak ties: A network theory revisited. In *Sociological Theory (1)*, 1983.

[15] L. Gravano, H. Garcia-Molina, and A. Tomasic. Gloss: Text-source discovery over the internet. *ACM Transactions of Database Systems*, 24(2):229–264, 1999.

[16] J. Kleinberg. The small-world phenomenon: An algorithmic perspective. In *Proc. 32nd ACM Symposium on Theory of Computing (STOC)*, pages 163–170, 2000.

[17] G. S. Manku. Routing networks for distributed hash tables. In *Proc. 22nd ACM PODC*, 2003.

[18] G. S. Manku, M. Bawa, and P. Raghavan. Symphony: Distributed hashing in a small world. In *Proc. 4th USENIX Symposium on Internet Technologies and Systems (USITS)*, pages 127–140, 2003.

[19] S. Milgram. The small world problem. In *Psychology Today 1(67)*, 1967.

[20] S. Milliner, M. Papazoglou, and H. Weigand. Linguistic tool based information elicitation in large heterogeneous database networks. In *Proc. Workshop on Natural Language and Databases (NLDB)*, 1996.

[21] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, June, 1995.

[22] C. H. Ng and K. C. Sia. Peer clustering and firework query model. In *Poster in 11th Conf. on World Wide Web (WWW)*, 2002.

[23] J. J. Ordille and B. P. Miller. Distributed active catalogs and meta-data caching in descriptive name services. In *Proc. Conf. on Distributed Computing Systems (ICDCS)*, pages 120–129, 1993.

[24] M. P. Papazoglou, H. A. Proper, and J. Yang. Landscaping the information space of large multi-database networks. *Data Knowledge Engineering*, 36(3):251–281, 2001.

[25] S. Ratnasamy, P. Francis, M. Handley, and R. M. Karp. A Scalable Content-Addressable Network (CAN). In *Proc. ACM SIGCOMM*, pages 161–172, 2001.

[26] H. Schutze and C. Silverstein. Projections for efficient document clustering. In *Proc. 20th ACM Conf. on Information Retrieval (SIGIR)*, pages 74–81, 1997.

[27] M. Sheldon, A. Duda, R. Weiss, Jr. O'Toole, and D. J. Gifford. A content routing system for distributed information systems. In *Proc. 4th Intl. Conf. on Extending Database Technology (EDBT)*, pages 109–122, 1994.

[28] L. Si, R. Jin, J. Callan, and P. Ogilvie. A language modeling framework for resource selection and results merging. In *Proc. 11th ACM Conf. on Information and Knowledge Management (CIKM)*, pages 391–397, 2002.

[29] P. Simpson. Query processing in a heterogeneous retrieval network. In *Proc. 11th ACM Conf. on Information Retrieval (SIGIR)*, pages 359–370, 1988.

[30] I. Stoica, R. Morris, D. Karger, M. Frans Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. ACM SIGCOMM*, pages 149–160, 2001.

[31] C. Tang, Z. Xu, and M. Mahalingam. Peersearch: Efficient information retrieval in peer-to-peer networks. In *HotNets-I*, 2002.

[32] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, 1979.

[33] C. L. Viles and J. C. French. Dissemination of collection wide information in a distributed information retrieval system. In *Proc. 18th ACM Conf. on Information Retrieval (SIGIR)*, pages 12–20, 1995.

[34] E. Voorhees, N. Gupta, and B. Johnson-Laird. Learning collection fusion strategies. In *Proc. 18th ACM Conf. on Information Retrieval (SIGIR)*, pages 172–179, 1995.

[35] R. Weiss, B. Velez, M. A. Sheldon, C. Nemprempre, P. Szilagyi, A. Duda, and D. K. Gifford. Hypursuit: A hierarchical network search engine that exploits content-link hypertext clustering. In *Proc. 7th ACM Conf. on Hypertext*, pages 180–193, 1996.

[36] J. Xu and J. Callan. Effective retrieval of distributed collections. In *Proc. 21st ACM Conf. on Information Retrieval (SIGIR)*, pages 112–120, 1998.

[37] J. Xu and W. B. Croft. Cluster-based language models for distributed retrieval. In *Proc. 22nd ACM Conf. on Information Retrieval (SIGIR)*, pages 254–261, 1999.

[38] B. Yang and H. Garcia-Molina. Improving search in peer-to-peer networks. In *Proc. 22nd Intl. Conf. on Distributed Computing Systems (ICDCS)*, 2002.