A Case for Associative Peer to Peer Overlays

Edith Cohen
AT&T Labs–Research
180 Park Avenue
Florham Park, NJ 07932, USA
edith@research.att.com

Amos Fiat
School of Computer Science
Tel Aviv University
Tel Aviv 69978, Israel
fiat@cs.tau.ac.il

Haim Kaplan
School of Computer Science
Tel Aviv University
Tel Aviv 69978, Israel
haimk@cs.tau.ac.il

ABSTRACT

The success of a P2P file-sharing network highly depends on the scalability and versatility of its search mechanism. Two particularly desirable search features are scope (ability to find infrequent items) and support for partial-match queries (queries that contain typos or include a subset of keywords). While centralized-index architectures (such as Napster) can support both these features, existing decentralized architectures seem to support at most one: prevailing protocols (such as Gnutella and FastTrack) support partial-match queries, but since search is unrelated to the query, they have limited scope. Distributed Hash Tables (such as CAN and CHORD) constitute another class of P2P architectures promoted by the research community. DHTs couple index location with the item's hash value and are able to provide scope but can not effectively support partial-match queries; another hurdle in DHT deployment is their tight control the overlay structure and data placement which makes them more sensitive to failures.

Associative overlays are a new class of decentralized P2P architectures. They are designed as a collection of unstructured P2P networks (based on popular architectures such as gnutella or FastTrack), and the design retains many of their appealing properties including support for partial match queries, and relative resilience to peer failures. Yet, the search process is orders of magnitude more effective in locating rare items. Our design exploits associations inherent in human selections to steer the search process to peers that are more likely to have an answer to the query.

1. INTRODUCTION

Peer-to-peer (P2P) networks have become, in a short period of time, one of the fastest growing and most popular Internet applications. As for any heavily used large distributed source of data, the effectiveness of a P2P network is largely a function of the versatility and scalability of its search mechanism.

Peer-to-peer networks came to fame with the advent of Napster [23], a centralized architecture, where the shared items of all peers are indexed in a single location. Queries were sent to the Napster Web site and results were returned after locally searching the central index; subsequent downloads were performed directly from peers. The legal issues which led to Napster's demise exposed all centralized architectures to a similar fate. Internet users and the research community subsequently turned to decentralized P2P architectures, where the search index and query processing, as well as the downloads, are distributed among peers.

Existing decentralized architectures can be coarsely partitioned into two groups [27]: unstructured, where search is *blind* (independent of the query or its context) and structured, where search is *routed*. Prevailing decentralized P2P architectures are *unstructured*. One of these architectures is Gnutella [14] under which items are only indexed by the peer that cache them; search can be resolved only by probing these peers; and peers are probed using flooding (that typically cover about 1000 nodes). The recent wave of FastTrack [33]-based P2P architectures (Morpheus, Kazaa [20, 19]) incorporate improved design that allows for more efficient downloads (simultaneous from several peers and ability to resume after failure); and improved search (by designating some peers as search-hubs *supernodes* that cache the index of others).

A feature that undoubtedly contributes to the beaming success of these decentralized unstructured architectures is support for *versatile (partial-match) queries*: Shared items typically have meta-attributes describing their type and properties (e.g., title, composer, performer); the search supports partial-match queries that populate a subset of these fields and may contain typos. Another important feature of these architectures is their "loose" structure, with each particular peer being relatively dispensable; what makes the network overlay more resilient to failures and frequent joins and disconnects. On the flip side, unstructured architectures lack an important feature which Napster had offered: While popular items (current hit movies) can be located and downloaded fairly efficiently, P2P users seemed to have lost the ability to locate less-popular items (60's hits).

A different class of architectures that was proposed and promoted by the research community is decentralized structured P2P architectures [31, 28, 29, 35, 13, 18], commonly referred to as Distributued Hash Tables (DHTs). With DHTs, peers are required to store or index certain data items, not necessarily those items that these peers have contributed or interested in. Additionally, some hashing algorithm is used to identify the peers storing a given data item. The connections between different peers are also a function of the architecture. Thus, while DHTs can be very effective for applications where queries involve unique item identifiers (e.g., P2P Web caching), they require that peers store data for the "common good"; they incur much larger overhead than "unstructured" architectures when peers fail or leave the network; and inherently, they can not efficiently support partial-match queries.

Associative overlays, proposed here, are decentralized P2P architectures, which on one hand, retain the desirable properties of prevailing unstructured architectures, including being fully decentralized with "loose" structure, and supporting partial-match queries, and on the other hand, address their biggest drawback by boosting the efficiency of locating infrequent items. Another desirable property of associative overlays is that peers are not required to store arbitrary data; peers store only what they use and their actions, including answering queries, have direct self benefit.

1.1 Associative overlays

Associative overlays defines both the formation of the overlay and the search process so that queries can be steered to peers that are more likely to have an answer. The basic premise, which we substantiate in the sequel, is that peers that would have been able to satisfy previous queries by the originating peer are more likely candidates to answer a current query.

Main ingredients in our design are *guide-rules* and *guided search*. A *guide rule* is a set of peers that satisfy some predicate; each peer can participate in a number of guide-rules, and for each guide-rule it participates in it maintains a small list of other peers belonging to the same guide rule. For each rule, the overlay induced by peers that participate in the rule forms an unstructured network and exhibits similar connectivity and expansion properties. *Guided search* retricts the propagation of queries to be within some specified guide-rules. When a peer originates a search for an item, it restricts the search propagation to a subset of its guide-rules. A peer propagating a search can only propagate it to neighbor peers within the specified rule(s).

Guide-rules should be such that peers belonging to some guide rule contain data items that are semantically similar, *e.g.*, contain documents that deal with the philosophy of science, or contain song titles by the artist formerly known as Prince. Guided search can be viewed as a middle ground between blind search used by unstructured networks and the routed search deployed by DHTs: Guided search provides a mechanism to focus the search, that is, the relevance of the peers that the query is propagated to, without tight control of the overlay and item locations. The search process within a rule mimicks search in unstructured networks, by essentially performing a blind search. On the other hand, the search strategy of the originating peer has the flexibility of deciding which guide rules, among those that the originating peer belongs to, to use for a given search.

The particular choice of the underlying set of guide-rules is constrained by both "networking" aspects, which require that the overlay has certain connectivity properties and can be formed and maintained at low cost, and the "data mining" aspects, which require that these rules meaningfully distill common interests; and thus, restricting the propagation of the query to peers within the guide rules of the originating peer yields a more focused search.

1.2 Possession rules

We focus on automatically-extracted guide rules of a very particular form, which we call *possession rules*. Each possession rule has a corresponding data item, and its predicate is the presence of the item in the local index, thus, a peer can participates in a rule only if it shares the corresponding item. Our underlying intuition, taken from extensive previous research in the Data-Mining and Text Retrieval communities ([17, 9, 10, 8, 5, 22, 16]), is that, on average, peers that share items (in particular rare items) are more likely to satisfy each other's queries than random peers. More precisely, search using possession-rules exploits presence of pairwise co-location associations between items.

Beyond the resolution of the search, possession rules provide an easy way to locate many other peers that share the item. This feature is useful for distributing the load of sending large files (parallel downloads are already practiced in FastTrack networks), or locating alternative download sites when a peer is temporarily swamped.

A feature that can allow associative overlays to strive under "selfish" peer behavior is that participation in guide-rules serves dual purpose: Supporting propagation of search through the peer but also allowing the peer to focus its own search process; A peer can participate in a rule only if it shares the corresponding item, and peers that do not participate in rules can not search better than via blind search.

1.3 The Rapier Search Strategy

The Rapier strategy is based on the following intuition: let the areas of interest for a given peer be A, B, C, *etc.*, randomly choose one of these areas of interest and perform a blind search amongst those peers that also have interest in this area. Rapier (Random Possession Rule) selects a possession-rule uniformly at random from the list of previously-requested items by the querying peer.

Evidently, if there are no correlations between items, Rapier has no advantage over blind search. We use a two-pronged evaluation of Rapier: First, we use a simple intuitive data model (the Itemset model) to learn how the effectiveness of Rapier grows with the amount of "structure" in the data. Second, we evaluate Rapier on actual data, using large datasets of users accessing web sites. We obtained that Rapier is likely to perform significantly better than blind search, in particular, it can be orders of magnitude more effective in searching for infrequent items.

2. RELATED WORK

The effectiveness of blind search can be boosted by aggregation and replication; for example, by peers summarizing the content available from other peers such as with super-peer architectures [33] and routing-indices [15] or by balancing the number of replicas against the query rates [12, 27]. The drawbacks of aggregation and proactive replication is that they are more sensitive to malicious or selfish peer behaviour and spreading of mis-labeled files. Associative overlays offer an orthogonal approach which could be combined with aggregation but does not require it.

Associative overlays address a networking challenge using an approach that is supported and motivated by extensive previous research in the field of data-mining. The constraints of the P2P setting, however, make it fundamentally different than traditional data-mining applications. A related classic data-mining problem is the *Market-basket problem*, which assumes a large number of items and customers that fill their baskets with some subset of the items. This framework applies to many domains of human activity including supermarket shopping (customers vs items matrix), library checkouts (readers vs books), document classification (word/terms vs documents matrix), Web page hyperlinks (Web pages vs Web pages), Web browsing (Users vs Web pages), and in our context, P2P networks (peers vs items). Common to all these datasets is the presence of structure in data, namely, that these matrices are far from random. It had been long recognized that these human-selection datasets are in a sense very structured [17, 24, 6, 1]

One purpose of market-basket mining is extracting *Association rules* [2, 3]. An example of an association rule is pairs of items that are often purchased together such as "Champaign and Caviar" or "Beer and Diapers." Such rules had been used for marketing (e.g., placing Beer and Diapers next to each other in the supermarket) and recommendation systems (e.g., recommend books to customers based on previous book purchases) [7, 21, 25, 4]. A computationally challenging important subproblem is to discover association rules that have *high correlation* but *low support* (e.g., the association rule "Champaign and Caviar" that are rare purchases but are often purchased together) [11].

Similarly to these data-mining techniques, we exploit the presence of associations; but the basic difference is our highly distributed setting. Our solution does not (and can not) explicitly obtain association rules but does heavily utilize their presence. Instead of clustering peers into communities we restrict the search to communities without explicitly identifying them.

Recent proposals to exploit "interest locality" to optimize p2p search also include [30], where an existing p2p network is extended by nodes linking directly to nodes that satisfied previous queries; This basic approach does not provide a mechanism to "focus" query propagation beyond the first hop. At the other end of the spectrum, PeerSearch [32], attempts to import traditional vector space Information Retrieval (at the cost of tightly controlled DHT overlay and communication overhead).

3. MODEL AND METHODOLOGY

We represent the data present in the network by the *peer-item* matrix $D \in \{0,1\}^{n\times m}$ where n is the number of peers, m is the number of items, and $D_{ij} = 1$ if and only if peer i contains data item j.

We define the *support set* of the jth item $S_j \subseteq \{1,\ldots,n\}$, $1 \le j \le m$, to be

$$S_j = \{\ell | D_{\ell j} = 1\}.$$

I.e., S_j is the set of all row indices (peers) that contain data item j. The *joint support set* of two items j, k,

$$S_{jk} = S_{jk} = \{\ell | D_{\ell k} = 1 \text{and } D_{\ell j} = 1\} \ ,$$

is the set of peers that contain both items. We refer to $X_i = \{j | D_{ij} = 1\}$ (the set of items associated with peer i) as the *index of peer i*. We use the notation $s_j = |S_j|$, $s_{jk} = |S_{jk}|$, and $x_i = |X_i|$.

We define $W_i = \frac{x_i}{|D|}$, where $|D| = \sum_{i=1}^{n} x_i$ is the combined size of all indexes. An item j has low support (is "rare") if $|S_j|/n$ is small. An item has low support with respect to the weights if $\sum_{i\in S_j} W_i \ll 1$.

We view the peer-item matrix as a current instantiation of the data. We measure performance of different algorithms by treating each "1" entry, in turn, as the most recent request: For each peer i and item j such that $D_{ij} = 1$, we refer to the request that corresponds to the i, j entry as the *query* (i, j). Each query triggers a search process, which depends on the matrix D with the entry D_{ij} set to 0 and on the peer i.[1] The search process is a sequence of probes: when a peer is probed, it attempts to match the query against its local index using some algorithm. We assume that this algorithm is perfect in the sense that a query of the form (i, j) can always (and only) be resolved by a probe to peer that contains the item j.[2] The size of a search process is a random variable, and the *Expected Search Size* Ess_{ij}^{A} is the expectation of this random variable.

We compare different strategies by looking at all *queries* (peer-item pairs with $D_{ij} = 1$). We sweep a threshold on the maximum value of the Ess, and look at the cumulative fraction of queries (i, j) that have Ess_{ij} below a threshold.

3.1 Blind Search as Random Search

Following [12, 27] we model the performance of blind search in "traditional" unstructured networks using the *Random Search* model. The intuition of why this abstraction is valid is that the set of probed peers on a query in unstructured networks depends only on the overlay structure which is independent of the query or previous selections by the querying peer. Thus, on average, the effectiveness of each probe can not be better than that of probing a random peer.

When comparing Rapier to blind search and to each other we must ensure that we do not compare apples and oranges. Rapier is somewhat biased towards searching in peers with relatively many items. Thus, comparing Rapier a blind search that chooses peers uniformly at random would be unfair. One might suspect that the advantages shown experimentally are due to the choice of peers with many items, and does not reflect any other property of these algorithms. To avoid this potential pitfall, we seek to ensure that we compare these algorithms to blind search algorithms that compete on equal terms. Specifically, we consider weighted versions of the random search model where hosts have different likelihood of receiving a probe: Each peer i has a weight w_i such that $\sum_i w_i = 1$, and the likelihood that a peer is visited in a random search probe is proportional to w_i. Weighted random search is used as a benchmark for the performance of our associative search algorithms. To obtain a fair comparison, we need to consider weights that reflect the bias of the associative search algorithms towards peers with larger index sizes.

We shall consider two natural weighting schemes:

- Uniform Random Search (Urand) where all peers are equally likely to be probed ($w_i = 1/n$). This models pure blind search.

- Proportional Random Search (Prand), where the likelihood that a peer is probed is proportional to the size of its index $w_i = W_i \propto \sum_{j=1}^{m} D_{ij}$. This models blind search biased towards peers with larger indices. We will show that this bias is exactly equal to the bias introduced by Rapier and thus differences in performance between the two cannot be due to this bias.

[1] Note that the search sequence does not depend on j, as query properties (such as meta-data terms) are not used to determine where to search. It is used only as a stopping condition. See the introduction and conclusion sections for a discussion on this issue.

[2] this simplification is justified as the matching issue of queries to appropriate items is present with other architectures and is orthogonal to the core of our contribution.

With weighted random search, the size of the search for a query (i, j) is a Geometric random variable. The ESS is the mean of this random variable.

A weighted random search for item j by peer i has likelihood of success in each probe $p_{ij} = \frac{\sum_{k \neq i} w_k D_{kj}}{1-w_i}$. and thus for any weighted random search algorithm A $\text{Ess}_{ij}^{A} = p_{ij}^{-1} = \frac{1-w_i}{\sum_{k \neq i} w_k D_{kj}}$. (The search is performed on all peers excluding peer i).

Thus, a URAND search for item j by peer i has

$$\text{Ess}_{ij}^{\text{URAND}} = \frac{n-1}{\sum_{k \neq i} D_{kj}} \; ; \tag{1}$$

and a PRAND search has

$$\text{Ess}_{ij}^{\text{PRAND}} = \frac{1-W_i}{\sum_{k \neq i} W_k D_{kj}} \; . \tag{2}$$

4. POSSESSION-RULE OVERLAYS

Guide rules connectivity. : Peers that participate in the same guide-rule form a sub-overlay that resembles a "traditional" unstructured network. Thus, each guide-rule constitutes a sub-overlay, and these sub-overlays are generally overlapping. Search is conducted using guide rules. Similarly to search in traditional unstructured networks, it is propagated from peer to neighbors but the propagation is only to peers belonging to the selected guide rule. Each guide-rule sub-overlay needs to have the form of a "traditional" unstructured overlay. For each guide-rule it is associated with, a peer needs to remember a small list of peers which belong to the guide rule; and neighbors should be such that guided-search reaches a large number of peers. The specifics can vary from a Gnutella-like design where each peer has few viable neighbors (Typical Gnutella number is 2-4) and many other peers can be reached through them, to a FastTrack-like design where search is facilitated through a core network of supernodes (in our case supernodes are associated with guide-rules). The specifics are orthogonal to our basic approach, we only need to make sure that our selected guide rules are such that the underlying unstructured network can form.

Search strategy. : A search strategy defines a search process as a sequence of guide rule probes. An example of a strategy is "search 100 peers that have item A and 200 peers that have item B, if this is unsuccessful, then search 400 more that have item A and 50 peers with item C, ..."

Our general expectation is that the total number of guide rules may be large, but a typical peer uses a bounded number of rules. The applicability of a specific set of guide-rules depends on the implementability of the connectivity requirement. This requirement has two parts, first there should be a simple mechanism to locate a peer (and through it other peers) that belong to the same guide rule. It is also a requirement that this selection should result in large connected components. Below we argue that possession-rules fill the first part. As for large components, practice shows that simple neighbor selection strategies of current P2P implementation result in large connected components, and thus, we argue that selections within a guide-rule are likely to result in large components. (Random connections are known to yield large components and apparently actual selections are "sufficiently random" to obtain this property). In any case, the same issue of obtaining large components exists in traditional unstructured architectures and the connectivity algorithms deployed in these networks can be adapted to our context. There is thus no need to re-tackle this issue.

The possession-rule overlay is self-boosting: If peer-A conducts a search for item i that is resolved by peer-B then it is able to obtain through peer-B a list of other peers that index item i. As a result, each peer has a neighbor list which is an array of (item,peer) pairs for (most) items in its index. Thus, for possession rules, the construction of the overlay is symbiotic with the search process. There is seemingly a major issue in that a peer in a guide-rule network may keep track of many other peers, proportional to the number of guide rules it belongs to. Even when bounding the number of guide-rules a peer participates in, the number of neighbors is considerably larger than in existing architectures. This is in contradiction to the philosophy used by existing P2P architectures, which promotes having a small number of neighbors. We argue, however, that there is no reason for guided search to abide by this rule whereas there are clear reasons for other P2P architectures to keep it. Unlike DHTs, the update cost of a neighbor going offline is minimal; we may discover it when trying to search through these peers and may then remove them from our list following one or more unsuccessful tries; replacements are easy to find if at least some of the guide-rule neighbors are active. It is also advantageous for search in unstructured network to have a small fan-out, but we achieve that since each guide-rule sub-overlay has a low degree.

In the sequel, we assume that our network is a possession-rule overaly. Each sub-overlay resembles an unstructured network and we use the model of random search used in [12, 27] to capture the performance of search within a rule.

5. RAPIER SEARCH STRATEGY

RAPIER is a simple search strategy that uses possession-rules overlay. The strategy repeats the following until search is successful (or search size limit is exceeded):

1. Choose a random item from your index.
2. Perform a blind search on the possession-rule for the item to some predetermined depth.

The main parameter we look at is the size of the search which is the total number of peers probed. We model RAPIER search by the following process: For a query for item j issued by peer i, a column k is drawn uniformly from $X_i \setminus \{j\}$ (the index of i excluding j). Then a peer r is drawn uniformly from $S_k \setminus \{i\}$. The search is successful iff $D_{rj} = 1$.

Thus, the likelihood of success for RAPIER per step is

$$p_{ij} = (x_i - 1)^{-1} \sum_{k \in X_i \setminus \{j\}} \frac{s_{kj} - 1}{n-1} \; .$$

and thus

$$\text{Ess}_{ij}^{\text{RAPIER}} = \frac{(x_i - 1)(n-1)}{\sum_{k \in X_i \setminus \{j\}} (s_{kj} - 1)} \; . \tag{3}$$

As discussed earlier, search strategies may differ to the extent that they utilize peers of different index sizes. RAPIER, in particular, is more likely to probe peers with larger indices, since such peers share items with a larger number of other peers. We can show that averaged over queries, the likelihood that a peer is probed under RAPIER is equal to W_i (its likelihood to be probed under PRAND). Thus, it is fair to use PRAND as a benchmark for RAPIER since *per-search*, they have the same bias towards peers with larger index sizes. We compare the performance of the two algorithms on the Itemset model and using simulations.

6. THE ITEMSETS MODEL

Frequency and size distributions of items and peers are reasonably-well understood and are typically modeled by Zipf-like distributions. But even though these distributions capture enough aspects of the data to evaluate the performance of blind search, they do not capture correlations that are necessary for evaluating associative search. Models which capture correlations present in market-basket data and Web hyperlink structure had been proposed [3, 25, 26]. We use one such model, the *Itemsets model* (which resembles models in [3, 25]), to convey intuition why and when we anticipate RAPIER to perform well.

The Itemsets model partitions items into N "interest areas" (which we refer to as *itemsets*). Each peer belongs to some subset of the itemsets, and contains f fraction of items (picked uniformly at random) in each itemset it belongs to.

Items in different itemsets are generally not correlated, and items in the same itemset are correlated. Our expectation is that if peers belong to many itemsets (at the extreme, all peers have all itemsets), there is no advantage for RAPIER over PRAND. When peers belong a to a small number of itemsets we expect RAPIER to perform better; and we expect this advantage to increase as the number of itemsets decreases. We formalize this intuition below.

Suppose that each peer belongs to exactly k itemsets[3], and these itemsets are independent or positively correlated, that is, if $p(x)$ is the fraction of peers belonging to itemset x, and $p(x \cap y)$ is the fraction of peers belonging both to itemsets x and y, then $p(x \cap y) \geq p(x)p(y)$. Let $x(\ell)$ be the itemset of item ℓ and let $p(x(\ell))$ be the fraction of the peers that contain itemset $x(\ell)$. Consider a query made to an item ℓ. Then the success probability of a PRAND probe is $R_\ell = fp(x(\ell))$; and the success probability of RAPIER probe is $C_\ell = \frac{f}{k}(1 + (k-1)p(x(\ell)))$. It follows that the ratio of the ESS under PRAND to the ESS under RAPIER for item ℓ by any peer is $\frac{1}{kp(x(\ell))} + \frac{k-1}{k}$. Since $p(x(\ell)) \leq 1$, RAPIER is always at least as effective as PRAND. When $p(x(\ell)) \ll 1/k$, RAPIER is much more efficient than PRAND. This simplistic model provides some intuition to when RAPIER is more effective than PRAND: RAPIER benefits, when users interests are more "focused" (small k) and for items in rare itemsets (small $p(x(\ell))$).

7. SIMULATION RESULTS

As large scale peer-item data is not available publicly, we opted to use a different source of similarly-structured ("market-basket") data. We used Boeing [34] Web proxy logs of a lower-level proxies serving end users and extracted the matrix of users versus hostnames. In the sequel, we refer to users as *peers* and to hostnames as *items*. The resulting data matrices (for each day of the Boeing logs) had about 57K peers, 45K items, and 115K pairs.

[3]Similar results would hold when we assume that each peer belongs to at most k itemsets

As is typical with such data, we observed high skew in both the size of the index peers have and the support-size of items (large fraction of peers having small index sizes and large fraction of items being present at a small fraction of peers. About 60% of queries are issued to items whose support is over 0.01 fraction of peers; so considerable fraction (40%) of queries target unpopular items.

We evaluated the performance of 3 search strategies:

Algorithm	ESS computed according to
URAND	Equation 1
PRAND	Equation 2
RAPIER	Equation 3

The results of the simulations are shown in Figures 1. The figure shows a cumulative fraction of queries that have ESS below a certain threshold. They show the performance for items across support levels and also focus on items that have lower support (occur in the index of at most 10^{-2}-10^{-4} of peers). The figures show that URAND is the worst performer. The ESS of URAND on an item is the inverse of its fraction of peers that index it, thus, when focusing on items occurring in at most 10^{-4} of users, the respective ESS is over 10K, and the URAND curve coincides with the x-axis. The PRAND strategy that prefers peers with larger index sizes manages to see more items in each probe and performs considerably better than URAND, across items of different support levels.

We observe that RAPIER, which has the same bias towards peers with larger index as PRAND, outperform PRAND; moreover, the performance gap is significantly more pronounced for items with low support. This indicates strong presence of the semantic structure RAPIER is designed to exploit; and also emphasizes the qualitative difference between RAPIER and aggregation-based search startegies.

For a typical Gnutella search size, estimated to cover about 1000 peers, the simulations on the Boeing dataset show that RAPIER covers 52% of queries made to items that are present on at most 10^{-4} fraction of peers, whereas PRAND covers only 14% of queries. Out of all queries, RAPIER covers 95% and PRAND covers 90%. On a smaller search size of a 100, RAPIER and PRAND, respectively, cover 30% and 1.3% of items with support below 10^{-4} fraction of peers, and cover 90% and 80% of all items. For search sizes where PRAND covers most queries, RAPIER obtains about half the failure rate of PRAND.

8. CONCLUSION

Associative overlays retain the advantages of unstructured architectures (such as gnutella and FastTrack); including relative insensitivity to peer failures and support for partial-match queries; but can offer orders of magnitude improvement in the scalability of locating infrequent items. Our design exploits presence of associations in the underlying data. Such associations were previously exploited for Web search, Data-mining, and collaborative filtering applications, but the techniques were not portable to the P2P setting which requires simple, resilient, and fully decentralised protocols. Our approach maintains the essense of these techniques while striking a balance with the challenges of the P2P setting.

We argued that RAPIER, the simplest search strategy on possession-rule overlays, can dramatically increase the effectiveneess of search for rare items over that of plain unstructured networks. It is likely that better search performance on possession-rule overlays can be achieved by prefer-

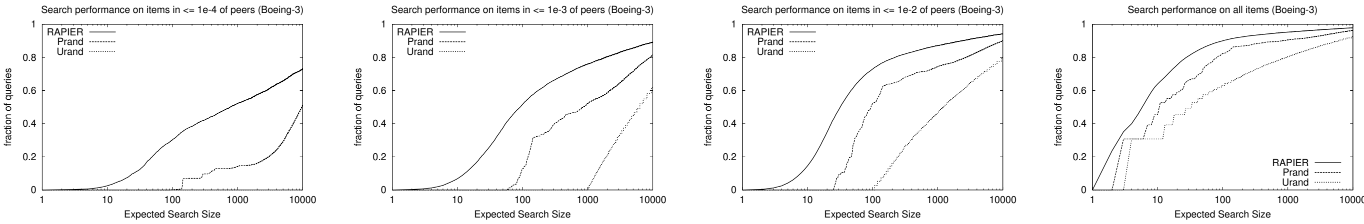

Figure 1: Search performance on items present in (1e-4, 1e-3, 1e-2, all) fraction of peers (Boeing-Day3 log).

ing rules that correspond to recently acquired items or rules where the meta data of the corresponding items is more related to the query terms. It is also possible to design more refined search strategies that acount for relations between guide rules.

9. REFERENCES

[1] D. Achlioptas, A. Fiat, A. Karlin, and F. McSherry. Spectral analysis of data. In *Symposium on Theoretical Computer Science, pp. 619-626*, 2001.

[2] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., 26–28 1993.

[3] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In J. B. Bocca, M. Jarke, and C. Zaniolo, editors, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, 1994.

[4] Y. Azar, A. Fiat, A. R. Karlin, F. McSherry, and J. Saia. Web search through hub synthesis. In *Symposium on Foundations of Computer Science*, 2001.

[5] K. Bharat and M. R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Research and Development in Information Retrieval*, pages 104–111, 1998.

[6] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the WWW7 Conference*, 1998.

[7] ACM SIGGROUP resource page on collaborative filtering. `http://www.acm.org/siggroup/collab.html`.

[8] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, S.R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Hypersearching the web. *Scientific American*, June 1999.

[9] S. Chakrabarti, B. Dom, D. Gibson, R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Spectral filtering for resource discovery, 1998.

[10] S. Chakrabarti, B. Dom, P. Raghavan, S. Rajagopalan, D. Gibson, and J. Kleinberg. Automatic resource compilation by analyzing hyperlink structure and associated text. *Computer Networks and ISDN Systems*, 30(1–7):65–74, 1998.

[11] E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J. D. Ullman, and C. Yang. Finding interesting associations without support pruning. *Knowledge and Data Engineering*, 13(1):64–78, 2001.

[12] E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks. In *Proceedings of the ACM SIGCOMM'02 Conference*, 2002.

[13] Open Source Community. The free network project - rewiring the internet. In *http://freenet.sourceforge.net/*, 2001.

[14] Open Source Community. Gnutella. In *http://gnutella.wego.com/*, 2001.

[15] A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer systems. In *ICDE*, 2002.

[16] J. Dean and M. R. Henzinger. Finding related pages in the world wide web. *WWW8 / Computer Networks*, 31(11-16):1467–1479, 1999.

[17] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic indexing. *Journal of the American Society for Information Science*, 41(6):391–407, September 1990.

[18] A. Fiat and J. Saia. Censorship resistant peer-to-peer content addressable networks. In *Symposium on Discrete Algorithms*, 2001.

[19] KaZaA file sharing network. KaZaA. In *http://www.kazaa.com/*, 2002.

[20] Morpheus file sharing system. Morpheus. In *http://www.musiccity.com/*, 2002.

[21] D. Goldberg, D. Nichols, Oki B. M., and D. Terry. Using collaborative filtering to weave an information tapestry. In *Communications of the ACM, 35(12)*, pages 51–60, 1992.

[22] T. Hofmann. Probabilistic latent semantic indexing. In *Research and Development in Information Retrieval*, pages 50–57, 1999.

[23] Napster Inc. The napster homepage. In *http://www.napster.com/*, 2001.

[24] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[25] S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Recommendation systems: A probabilistic analysis. In *IEEE Symposium on Foundations of Computer Science*, pages 664–673, 1998.

[26] S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Extracting large-scale knowledge bases from the web. In *The VLDB Journal*, pages 639–650, 1999.

[27] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th annual ACM International Conference on supercomputing*, 2002.

[28] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proceedings of SIGCOMM'2001*, 2001.

[29] A. Rowstron and P. Druschel. Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility. In *Proceedings of SOSP'01*, 2001.

[30] K. Sripanidkulchai, B. Maggs, and H. Zhang. Enabling efficient content location and retrieval in peer-to-peer systems by exploiting locality in interests. *ACM SIGCOMM Computer Communication Review*, 32(1), January 2002.

[31] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of SIGCOMM'2001*, 2001.

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

[33] FastTrack Peer-to-Peer technology company. FastTrack. In *http://www.fasttrack.nu/*, 2001.

[34] Web traces and logs. `http://www.web-caching.com/traces-logs.html`.

[35] B. Y. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, University of California at Berkeley, Computer Science Dept, 2001.