

GRaSP: Generalized Range Search in Peer-to-peer Networks

Michail Argyriou
Technical University of Crete
micharg@intelligence.tuc.gr

Vasilis Samoladas
Technical University of Crete
vsam@softnet.tuc.gr

Spyros Blanas
Univ. of Wisconsin at Madison
sblanas@cs.wisc.edu

ABSTRACT

We present a framework for generalized range search on trie-structured P2P networks, such as P-Grid. Our techniques exploit hitherto unknown properties of randomized tries. We prove that a P-Grid like network has routing diameter $O(\log n)$ with high probability, as well as $O(\log n)$ congestion, regardless of the shape of the underlying trie. Based on these properties, we propose GRaSP, a simple scheme for handling arbitrary range search problems, with search and update hop latency $O(\log n)$ with high probability. We then apply GRaSP on two range search problems: multidimensional range search over points and rectangles, and three-sided search. Our empirical results show that GRaSP delivers excellent search performance and exhibits very good scalability under heavy load. With respect to three-sided search, our proposed scheme is distinguished in that it attempts to improve load balancing by introducing redundancy via the choice of search space.

1. INTRODUCTION

Driven by the need for scalable, decentralized data management in advanced applications, support for range search in P2P networks is currently under intensive study. Much of the relevant literature focuses on application specific techniques, including single-attribute range queries, spatial range queries, nearest-neighbor search, multi-attribute queries over application-specific data. There are relatively fewer proposals for techniques that can generalize to arbitrary search problems. The main difficulty in generalizing application-specific techniques is that the features of particular search problems (dimensionality, skew, data and query distribution) are often exploited to improve performance and scalability.

Of particular interest in the literature is the problem of adapting hierarchical data structures and indexing schemes of low height, such as search trees, in the P2P domain. Adapting hierarchical structures to P2P networks is challenging: if done naively, the peers corresponding to the root

of the hierarchy can easily become overloaded. To overcome this difficulty, two families of techniques have been proposed. The first family includes techniques which attempt to replicate the points of entry to the hierarchy; many of these techniques are adaptations of randomized data structures, such as skip lists, and most works focus on single-attribute range search [6, 23, 10, 16, 14]. The second family of techniques, exemplified by VBI-tree [21], avoids visiting the higher levels of the hierarchy, by performing some sort of “sideways” search. A common theme in both families of techniques is that they attempt to keep the search hierarchy relatively shallow, of height typically logarithmic to the size of the network.

Another approach is taken by techniques based on space partitioning. These techniques often combine a Distributed Hash Table (DHT) with an appropriate mapping of the search space to the space of hash keys. Such mappings are typically derived starting with an order-preserving hash function, suitable for single-attribute search, and handling multi-dimensional data by employing space-filling curves. A challenge in these techniques is to balance the data evenly among peers; some application-specific assumptions on the distribution of data are needed, in order to select the mappings appropriately.

Aberer and his collaborators proposed and studied P-Grid (e.g., [2, 11, 3, 1]), a DHT whose underlying structure can be described in terms of a binary trie. The distinguishing feature of P-Grid is that routing tables are constructed in a randomized, highly dynamic manner, with peers constantly exchanging neighbor information with other peers. The elegance and simplicity of P-Grid’s protocols is one of its strongest points, for pragmatic reasons.

The trie-like structure of P-Grid offers the tantalizing possibility of combining the DHT with hierarchical space partitioning schemes, avoiding order-preserving hashing and space-filling curves. Yet, previous work on range search over P-Grid has not explored this possibility. In fact, the properties of the underlying trie have not been studied analytically, with the exception of [1]; in that work, it is shown that the expected number of hops between two peers is logarithmic to the size of the network. This result is interesting, because it holds for tries of any shape, even for those degenerate shapes where every non-leaf node has a leaf child.

This paper presents new analytical performance results for trie-structured networks, which are applied to GRaSP, *Generalized Range Search over P-Grid*. On the analytical side, we extend the result of [1] by proving that the routing diameter (the length of the longest route) of any trie-based

P2P network is $O(\log n)$ with high probability (w.h.p.), regardless of the shape underlying trie. We complement this result with a $O(\log n)$ bound on congestion: when each peer routes a message to some other, randomly selected peer, we show that the expected number of messages going through any peer is $O(\log n)$, for any trie and regardless of the position of the peer in the trie.

Our analytical results are significant in themselves, in that they quantify the performance of PGrid as a DHT. But they also have a profound impact on the development of protocols for complex search over trie-structured networks, as they decouple routing concerns from the shape of the trie. We explore the case of generalized range search, introducing GRASP, a P2P framework which can be extended to particular range search problems with little effort, simply by providing a hierarchical space-partitioning function which maps binary strings to space ranges. In many ways, GRASP can be likened to Generalized Search Trees (GiST) [17], an index structure for generalized range search in databases.

To validate our techniques, we study experimentally two applications of GRASP; multidimensional range search, and 3-sided search. The latter is a special case of 2-d range search, where the queries consist of rectangles open in one direction. In this problem, we attempt to balance load by introducing data redundancy among the peers, through the chosen space partition. The experiments validate our claim that GRASP can be used to construct scalable networks for a broad range of applications.

2. RELATED WORK

Research in indexing on P2P networks started with the introduction of Distributed Hash Tables (DHTs) [27, 31, 29, 25], but soon extended to complex problems, such as multi-attribute indexing, e.g. [7, 18], nearest-neighbor and similarity indexing, e.g. [22, 15, 32], and range search in one dimension, e.g. [20, 6, 11, 23, 10] or in multiple dimensions, e.g. [21, 9, 12, 4, 33].

Peer-to-peer range search in one dimension can be done by modifying DHTs to use an order-preserving hash function [11, 23]. Most relevant to our work is the work of Datta et al. [11], which also employs P-Grid; their so-called *shower protocol* is similar to the **Search** protocol of GRASP, described in §3.4. That paper also presents experiments with an implementation on PlanetLab.

Multidimensional search over DHTs can be done by mapping the multidimensional space on the hash space, e.g. using space-filling curves. Ganesan et al. [12] propose two structures, SCRAP, based on space-filling curves over a skip graph-like network, and MURK, a CAN-derived network which partitions space in a manner similar to k -d trees. They evaluate their techniques experimentally, but do not consider congestion.

Other proposals are based on fault-tolerant, distributed variants of skip lists. Many of these works prove analytical guarantees for latency, congestion, update time, etc., but are not complemented by experimental evaluation. Also, the techniques are presented as distributed data structures, i.e., the complexity results are expressed with respect to the number N of data items stored in the network, not the number n of peers in the network, which can complicate matters when $N \gg n$. Protocols are also described at the level of data items, not the peers storing them. Yet, even with these practical limitations, these structures can pro-

vide strong performance guarantees. For one-dimensional search, Aspnes and Shah [6] propose skip graphs, and Harvey et al. [16] propose SkipNet. These structures exhibit query and update hop latency $O(\log N)$ w.h.p. and congestion $O(\log N)$. [Congestion is the maximum, over all nodes, of the expected number of queries visiting any node, when n uniformly distributed queries are executed; thus, for congestion c , the fraction of queries visiting a node is c/n]. Goodrich et al. [14] introduce rainbow skip graphs, providing essentially the same performance guarantees, but with additional guarantees for fault tolerance. For multidimensional queries, Arge et al. [4] propose SkipWebs, a structure that can handle a family of multidimensional problems (when some technical restrictions are satisfied), with expected $O(\log N / \log \log N)$ messages per query or update, and $O(\log N)$ congestion.

Another line of work, attempts to adapt tree-based data structures to P2P networks. In [20], Jagadish et al. describe BATON, a binary-tree like P2P network for 1-d range search, with $O(\log n)$ query cost (here, n is the number of peers) and amortized $O(\log n)$ update cost. They generalize this approach in BATON* [19], introducing a k -ary tree with $O(\log_k n)$ search cost, but an increased update cost of $O(k + \log n)$. BATON* is also able to cope with multi-attribute queries. In a subsequent paper, they generalize these ideas to VBI-tree, a binary-tree like network that can cope with general search problems, with a search latency of $O(\log n)$. They validate their work on an adapted version of the M-tree and nearest-neighbor queries.

3. TRIE-STRUCTURED NETWORKS

The goal in this section is to develop the necessary definitions needed in the description of GRASP protocols, and in the analytical results of the next section.

3.1 Notation

We shall use the following notation: a binary string x has length $|x|$, and ϵ denotes the empty string. We write $x \sqsubseteq y$ to denote that x is a prefix of y . The longest common prefix of x and y is denoted by $x \uparrow y$ and their concatenation by $x \cdot y$. Finally, $x[i]$, $0 \leq i < |x|$ is the i -th symbol of x (starting with 0), $x[:j]$ is the prefix of size j , $x[i:]$ is the suffix of size $|x| - i$ and $x[i:j] = x[i:] \cdot x[:j - i]$.

A prefix code is a finite set \mathcal{P} of (finite) binary strings, with the following property: for every infinitely long binary string x , there is a *unique* $y \in \mathcal{P}$ such that $y \sqsubseteq x$.

The i -th complement of x is $x[:i] \cdot \overline{x[i]}$ (for $i < |x|$).

3.2 Binary tries

A binary trie is a full binary tree, i.e., a binary tree where each non-leaf node has exactly two children. Each node n of the trie can be associated with a binary string $I(n)$, called the *node ID*, by the following rule: the node ID of the root is the empty string, and for every other node u , with parent v , if it is a left child of v then $I(u) = I(v) \cdot 0$, else $I(u) = I(v) \cdot 1$. A binary trie can be fully described by the set of node IDs of its leaves, which constitute a prefix code.

In the context of PGrid, any trie of n leaves gives rise to a network of n peers, where each peer is associated with a distinct leaf. Thus, we describe the shape of a trie-structured network by a prefix code \mathcal{P} . For simplicity, we often identify a peer with the node ID of the corresponding leaf (its peer

ID), that is, the elements of \mathcal{P} (which are binary strings) will be referred to as peers.

Let us fix a trie \mathcal{P} . For $p, q \in \mathcal{P}$, let

$$p \triangleright q = |p| - |p \uparrow q|$$

be called the gap from p to q . Note that $0 \leq p \triangleright q \leq |p|$. Informally, $p \triangleright q$ can be described in terms of the trie: starting from p , one must ascend $p \triangleright q$ nodes in the trie, before it can descend towards q . Thus, $p \triangleright q$ is, in a sense, a distance function; note however that it is not symmetric. The most useful law regarding gaps is:

Theorem 1 (Routing rule)

$$p \triangleright q < p \triangleright r \Leftrightarrow q \triangleright p < q \triangleright r \Leftrightarrow r \triangleright p = r \triangleright q$$

The proof is easy from the definition. Fig. 1 depicts this law graphically.

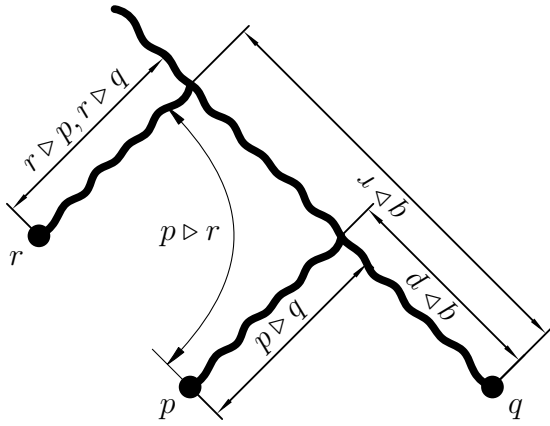


Figure 1: Sketch of a trie and three leaves, p, q, r , depicting the routing rule. Thick wavy lines denote trie paths from the root to the three leaves. Gaps are shown as trie path lengths.

3.3 Basic routing

In order to route messages among the peers, each peer maintains a set of pointers to other peers. For peer $p \in \mathcal{P}$, define a $(|p| + 1)$ -partition of the set of peers \mathcal{P} as follows:

$$N_i^p = \{q \in \mathcal{P} \mid p \triangleright q = i\} \text{ for } 0 \leq i \leq |p|.$$

Alternatively, each set N_i consists of the leaves of the subtree (of the trie) rooted at the trie node with node ID equal to

$$p[: |p| - i] \cdot \overline{p[|p| - i]}$$

(the $(|p| - i)$ -th complement of p).

The routing table of p is constructed by selecting *uniformly at random* one peer from each N_i^p . Let $L_i^p \in N_i^p$ denote the selected peer.

In order to route a message from p to q , the message is forwarded from p to $r = L_{p \triangleright q}^p$. From r it is recursively forwarded to $L_{r \triangleright q}^r$, and so on, until it reaches q . To see that this will happen, note that $p \triangleright r = p \triangleright q$, and by the routing rule, $q \triangleright p > q \triangleright r$: with each hop, the gap from the destination q to the current node decreases, down to 0.

3.4 Range search

Let U be an arbitrary set, called the *search space*. The elements of U are called points. A key space \mathcal{K} is a family of non-empty subsets of U , whose elements are called keys. A range space \mathcal{R} is also a family of non-empty subsets of U , whose elements are called spaces. A dataset $K \subseteq \mathcal{K}$ is a *finite* subset of the key space. Given a range $R \in \mathcal{R}$, and a dataset K , the answer $A_K(R)$ to R over K is the set of elements of K which intersect R :

$$A_K(R) = \{x \in K \mid x \cap R \neq \emptyset\}.$$

A *hierarchical binary space partition* is a function S from binary strings to subsets of U , such that $S(\epsilon) = U$, and for each string u , $\{S(u \cdot 0), S(u \cdot 1)\}$ is a partition of $S(u)$, i.e., $S(u \cdot 0) \cap S(u \cdot 1) = \emptyset$ and $S(u \cdot 0) \cup S(u \cdot 1) = S(u)$.

For a given space partition S , assign $S(p) \subseteq U$ to be the peer range of p . Since the peer IDs form a prefix code, it is easily seen that the peer ranges partition the search space.

Given a dataset K , each peer p stores $A_K(S(p))$. Note that this storage scheme implies redundancy; keys may be stored in multiple peers.

To answer a range query for range R , starting from some initial peer q , all that is needed is to forward the search to those peers whose range intersects R . These peers will collectively report the full answer to the range query.

The following protocol can be used to locate the relevant peers:

```

Search(peer  $p$ , range  $R$ , int  $l$ ) {
  if(  $R \cap S(p) \neq \emptyset$  ) answerLocally( $R$ );
  for(int  $i = l$ ;  $i < |p|$ ;  $i++$ )
    if(  $S(p[: i] \cdot \overline{p[i]}) \cap R \neq \emptyset$  ) Search( $L_{|p|-i}^p$ ,  $R$ ,  $i + 1$ );
}

```

where the search is initiated by a call to **Search**($q, R, 0$). In terms of a P2P network, **Search**(p, R, l) is a message sent to p . Parameter l is used to restrict the scope of search. It denotes that p should only forward the search to the part of the network corresponding to a subtree of the trie, rooted at the trie node with nodeID $p[: l]$. This subtree includes every peer q with $q \sqsubseteq p[: l]$. Peer p fulfills this request by forwarding further to each network subset $N_{|p|-i}^p$, for $l \leq i < |p|$. However, the search is pruned for those i where there will be no answer from the corresponding subtree. Routine **answerLocally**(R) poses range query R to the set of keys stored at peer p , and reports $A_K(S(p)) \cap A_K(R)$ to the user.

4. ANALYSIS OF TRIE-STRUCTURED NETWORKS

We now turn our attention to the cost of searching over tries. We are interested in two metrics: hop latency and congestion. Hop latency is the maximum distance (in terms of hops) from the initial peer to any peer reached during the search. The congestion at peer p is the number of messages forwarded via p , when every peer in the network routes a search to some other, randomly chosen peer.

4.1 Latency

Let us consider a fixed trie with n leaves. Let $H(p, q)$ be the number of hops in which a message is routed from p to

q . This quantity can be defined recursively as follows:

$$H(p, q) = \begin{cases} 0 & \text{if } p = q \\ H(L_{p \triangleright q}^p, q) + 1 & \text{if } p \neq q \end{cases} \quad (1)$$

For each p, q , $H(p, q)$ is a random variable, whose value depends on the random choices of L_i^r (the contents of the routing tables) of peers in the network.

To characterize $H(p, q)$ statistically, We will need the following simple lemma:

Lemma 1 For $p \neq q$,

$$N_{p \triangleright q}^p = \bigcup_{i=0}^{q \triangleright p - 1} N_i^q$$

PROOF.

$$r \in N_{p \triangleright q}^p \Leftrightarrow p \triangleright q = p \triangleright r \Leftrightarrow q \triangleright r < q \triangleright p \Leftrightarrow r \in \bigcup_{i=0}^{q \triangleright p - 1} N_i^q$$

□

Theorem 2 For any p, q ,

$$E[H(p, q)] \leq H_{|N_{p \triangleright q}^p|},$$

where H_k is the harmonic sequence.

PROOF. Let $k = q \triangleright p$. Define $n_i = |N_i^q|$ for $0 \leq i < k$, and let $s_i = \sum_{0 \leq j < i} n_j$. Notably, $n_0 = 1$ and $s_0 = 0$. From Lemma 1, it is clear that $|N_{p \triangleright q}^p| = s_k$, thus we must show that $E[H(p, q)] = H_{s_k}$.

Now,

$$\begin{aligned} E[H(p, q)] &= \sum_{r \in N_{p \triangleright q}^p} (E[H(r, q)] + 1) \Pr[L_{p \triangleright q}^p = r] \\ &= \sum_{i=0}^{k-1} \sum_{r \in N_i^q} (E[H(r, q)] + 1) \Pr[L_{p \triangleright q}^p = r] \\ &= \sum_{i=0}^{k-1} \sum_{r \in N_i^q} (E[H(r, q)] + 1) \frac{1}{s_k} \end{aligned}$$

It is easy to show from the above formula (by induction on k) that $E[H(p, q)]$ is the same for all peers p with $q \triangleright p = k$. In particular, define d_i to be equal to $E[H(r, q)]$ for any peer $r \in N_i^q$. Then,

$$d_k = \sum_{i=0}^{k-1} \frac{n_i}{s_k} (d_i + 1) = 1 + \sum_{i=0}^{k-1} \frac{n_i}{s_k} d_i.$$

To solve this recurrence, rewrite it as

$$s_k(d_k - 1) = \sum_{i=0}^{k-1} n_i d_i.$$

Then, for $k \geq 2$,

$$\begin{aligned} s_k(d_k - 1) &= \sum_{i=0}^{k-1} n_i d_i \\ &= n_{k-1} d_{k-1} + \sum_{i=0}^{k-2} n_i d_i \\ &= n_{k-1} d_{k-1} + s_{k-1} (d_{k-1} - 1) \\ &= s_k d_{k-1} - s_{k-1} \end{aligned}$$

By rearranging, we get

$$d_k = d_{k-1} + n_{k-1}/s_k$$

for $k \geq 2$. Note also that $d_1 = 1 = n_0/s_1$, thus,

$$\begin{aligned} d_k &= \sum_{i=0}^{k-1} \frac{n_i}{s_{i+1}} = \sum_{i=0}^{k-1} \sum_{j=1}^{n_i} \frac{1}{s_{i+1}} \\ &\leq \sum_{i=0}^{k-1} \sum_{j=1}^{n_i} \frac{1}{s_i + j} = H_{s_k}. \end{aligned}$$

□

The above theorem has already been shown in [1]. However, it does not allow us to claim that a range search has logarithmic cost, even on average; a range search may branch out to multiple destination peers, and the search cost is the maximum length along all routes. Thus, to claim average logarithmic search cost we would need to show that the expected routing diameter of the network is logarithmic. However, we can prove something stronger; that the routing diameter is $O(\log n)$, with high probability.

Theorem 3 For any trie \mathcal{P} of n leaves, and any $a > 0$,

$$\Pr[\max_{p, q \in \mathcal{P}} H(p, q) > (a + 3) \log n + 2] \leq 1/n^a$$

PROOF. First, we will show that $H(p, q)$ is $O(\log n)$ w.h.p. for fixed p, q . Then we will generalize over all pairs p, q .

For fixed p, q , let $k = q \triangleright p$. We number sequentially the peers in $N_{p \triangleright q}^p = \bigcup_{0 \leq i < k} N_i^q$, starting from 0, in such a way that if $\chi(r)$ denotes the numbering of peer r , then

$$q \triangleright r_1 < q \triangleright r_2 \Rightarrow \chi(r_1) < \chi(r_2),$$

that is, the peers are numbered in an order consistent with their gap from q . Also, let $\chi(p) = |N_{p \triangleright q}^p|$.

Now, let us write a probabilistic recurrence for $H(p, q)$, using our numbering to rewrite Eq. 1. Let $T(x) = H(\chi^{-1}(x), q)$. From the recursive form of $H(p, q)$, we have

$$T(x) = T(\hat{L}(x)) + 1 \quad (2)$$

where function $\hat{L}(x)$ is defined as follows: if $x = \chi(r)$, then $\hat{L}(x) = \chi(L_{r \triangleright q}^r)$.

The crucial observation is that $E[\hat{L}(x)] < x/2$. To see this, observe that (for $r = \chi^{-1}(x)$) $|N_{r \triangleright q}^r| < x$ and $\hat{L}(x)$ is uniformly distributed over $[0 : |N_{r \triangleright q}^r|)$.

Probabilistic recurrences like Eq. 2 are studied in [24], where it is shown that

$$\Pr[T(x) \geq \lceil \log x \rceil + w + 1] \leq (1/2)^{w-1} \quad (3)$$

for all positive real x and positive integers w .

Now, fix any $a > 0$, and let n be the number of peers in the network. Setting

$$w = \lceil (a + 3) \log n - \lceil \log n \rceil \rceil,$$

Eq. 3 becomes (after manipulations)

$$\Pr[H(p, q) \geq (a + 3) \log n + 2] \leq 1/n^{a+2}$$

Having obtained a tail bound on $H(p, q)$ for any p, q , we now have

$$\begin{aligned}
& \Pr\left[\max_{p, q \in \mathcal{P}} H(p, q) \geq (a+3) \log n + 2\right] \\
&= \Pr\left[\bigvee_{p, q \in \mathcal{P}} H(p, q) \geq (a+3) \log n + 2\right] \\
&\leq \sum_{p, q \in \mathcal{P}} \Pr[H(p, q) \geq (a+3) \log n + 2] \\
&\leq n^2 \frac{1}{n^{a+2}} = 1/n^a.
\end{aligned}$$

which concludes the proof. \square

The above result implies that, for any trie, regardless of the space partition, the latency of our **Search** protocol is a small multiple of $\log n$, with high probability.

4.2 Congestion

Regarding network congestion at peers due to concurrent searches, it does not seem feasible to prove as general a result as for latency; congestion depends on the choice of space partition and the distribution of arriving queries. For any space partition, if queries are skewed enough, hitting heavily on just a few peers, then, as the rate of arriving queries increases, these peers will soon become overloaded, reducing performance.

However, it is instructive to examine the case where searches access peers in a more or less uniform manner. Such an analysis would reveal whether there are any hotspots due to the shape of the trie, or to a bad (but likely) choice of routing tables.

We use a standard definition of congestion found in the literature. We consider the following communication pattern: each peer p selects uniformly at random another peer $C(p)$ from the network and routes a message to $C(p)$. Given that each of the n peers starts a process that will create $\Theta(\log n)$ messages w.h.p., the total number of messages is $\Theta(n \log n)$. We will prove that, for any trie and any peer p , the expected number of messages through p is $O(\log n)$.

To formalize the problem, define a set of indicator random variables $R(q, p, p')$, where $R(q, p, p')$ is 1 if the route from p to p' passes through q , and 0 otherwise. A first observation on the properties of $R(q, p, p')$ comes from routing:

Lemma 2 For $p \neq q$, $R(q, p, p') = 0$ unless $q \triangleright p > q \triangleright p'$

PROOF. If $q \triangleright p = q \triangleright p'$ then from the routing rule $p' \triangleright p < p' \triangleright q$. Also, if $q \triangleright p < q \triangleright p'$, then from the routing rule $p' \triangleright q = p' \triangleright p$.

But, for any peer $r \neq p$ on the route from p to p' , it must be $p' \triangleright r < p' \triangleright p \leq p' \triangleright q$. Thus, $r \neq q$. \square

Now, we can show the following:

Lemma 3 For any q, p, p' , such that $q \triangleright p > q \triangleright p'$,

$$E[R(q, p, p')] = \frac{1}{\sum_{i=0}^{q \triangleright p'} |N_i^q|}$$

PROOF. As in the proof of Theorem 2, define $n_i = |N_i^q|$ and let $s_i = \sum_{0 \leq j < i} n_j$. We need to show that

$$E[R(q, p, p')] = 1/s_{q \triangleright p' + 1}.$$

We have

$$\begin{aligned}
& E[R(q, p, p')] \\
&= \Pr[R(q, p, p') = 1] \\
&= \sum_{r \in N_{p \triangleright p'}^p} \Pr[R(q, r, p') = 1 | L_{p \triangleright p'}^p = r] \cdot \Pr[L_{p \triangleright p'}^p = r] \\
&= \sum_{i=0}^{q \triangleright p - 1} \sum_{r \in N_i^q} \Pr[R(q, r, p') = 1 | L_{p \triangleright p'}^p = r] \cdot \Pr[L_{p \triangleright p'}^p = r] \\
&= \sum_{i=0}^{q \triangleright p - 1} \sum_{r \in N_i^q} \frac{1}{s_{q \triangleright p}} \Pr[R(q, r, p') = 1 | L_{p \triangleright p'}^p = r]
\end{aligned}$$

However, for $0 < q \triangleright r \leq q \triangleright p'$, Lemma 2 implies $R(q, r, p') = 0$, thus, in the last sum above, the terms for $0 < i \leq q \triangleright p'$ are all zero. The term for $i = 0$ is $1/s_{q \triangleright p}$ (the probability that $L_{p \triangleright p'}^p = q$). Thus, the sum becomes

$$\begin{aligned}
& E[R(q, p, p')] = \\
& \frac{1}{s_{q \triangleright p}} + \sum_{q \triangleright p' < i < q \triangleright p} \sum_{r \in N_i^q} \frac{1}{s_{q \triangleright p}} \Pr[R(q, r, p') = 1 | L_{p \triangleright p'}^p = r]
\end{aligned}$$

Now we can proceed by induction. Letting $k = q \triangleright p$ and $l = q \triangleright p' < k$, the above sum must be shown equal to $1/s_{l+1}$. For the base case $k = l + 1$, the above formula becomes $1/s_k = 1/s_{l+1}$. For the inductive step,

$$\begin{aligned}
& \frac{1}{s_k} + \sum_{l < i < k} \frac{1}{s_k} \sum_{r \in N_i^q} \Pr[R(q, r, p') = 1 | L_{p \triangleright p'}^p = r] \\
&= \frac{1}{s_k} + \sum_{l < i < k} \frac{1}{s_k} \frac{n_i}{s_{l+1}} \\
&= \frac{s_{l+1} + \sum_{l < i < k} n_i}{s_k s_{l+1}} \\
&= \frac{s_k}{s_k s_{l+1}} = \frac{1}{s_{l+1}}.
\end{aligned}$$

completing the proof. \square

Now, consider the total traffic $F(q)$ going through peer q when each peer p forwards a message to $C(p) \neq p$. Since a peer is never visited twice for the same search,

$$F(q) = \sum_{p \in \mathcal{P}} R(q, p, C(p)).$$

Theorem 4 In a trie of n leaves, for any peer q ,

$$E[F(q)] \leq H_{n-1} + 1.$$

PROOF. From the definition of $F(q)$,

$$\begin{aligned}
E[F(q)] &= 1 + \sum_{q \neq p} \Pr[R(q, p, C(p)) = 1] \\
&= 1 + \sum_{q \neq p \neq p'} \Pr[R(q, p, C(p)) = 1 | C(p) = p'] \cdot \\
& \quad \Pr[C(p) = p'] \\
&= 1 + \frac{1}{n-1} \sum_{q \neq p \neq p'} E[R(q, p, p')]
\end{aligned}$$

Applying Lemma 2, we get

$$E[F(q)] = 1 + \sum_{p \neq q} \sum_{l=0}^{q \triangleright p - 1} \sum_{p' \in N_l^q} E[R(q, p, p')]$$

Let us define $n_i = |N_i^q|$ and $s_i = \sum_{0 \leq j < i} n_j$ as before. For $p' \in N_i^q$, from Lemma 3,

$$E[R(q, p, p')] = 1/s_{l+1}.$$

Letting $k = q \triangleright p$, we have

$$\begin{aligned} & \sum_{l=0}^{k-1} \sum_{p' \in N_l^q} E[R(q, p, p')] \\ &= \sum_{l=0}^{k-1} \frac{n_l}{s_{l+1}} \\ &\leq H_{s_k} \end{aligned}$$

where the last inequality was already shown in the proof of Theorem 2.

Thus,

$$\begin{aligned} E[F(q)] &\leq 1 + \frac{1}{n-1} \sum_{p \neq q} H_{|N_{p \triangleright q}^p|} \\ &\leq 1 + \frac{1}{n-1} \sum_{p \neq q} H_{n-1} \\ &= 1 + H_{n-1} \end{aligned}$$

concluding the proof. \square

5. NETWORK MAINTAINANCE

So far we have concentrated on describing and analyzing the performance of range search over a static P2P network. In most applications, P2P networks are in a continuous state of flux, as peers join and leave, often by failing. Also, new data arrives constantly and must be inserted into the network.

One of the most appealing features of GRaSP is that, as a straightforward extension of P-Grid, it inherits many of its protocols from it. P-Grid has been extensively studied in previous work, both by simulation and by implementation, and its performance is well documented. So, we shall only discuss those issues where GRaSP differentiates from P-Grid. These issues are, insertion and deletion of keys, and peer joins. Other issues, including the handling of failing peers, peers leaving the network gracefully, updating of routing tables etc. are handled identically to P-Grid.

5.1 Data updates

To insert a new key into the network, one can use the **Search** procedure of §3.4, where now in place of a search range we pass the actual key to be inserted. This key may be replicated to multiple peers. Deletions can be handled in a similar manner. For bulk updates, e.g., when a newly arriving peer wishes to index multiple keys, the same basic procedure applies, except that the set of indexed keys should be distributed at each forwarding step, in order to minimize the amount of data transferred over the network. Note that the hop latency of all these procedures is still $O(\log n)$ w.h.p.

5.2 Peer joins

A new peer p who wishes to join the network, must contact some existing peer whose network address is known to it, called the *bootstrap* peer. Then, it must select a *mate*, that is, an existing peer q (which can be the bootstrap itself), whose region it will split, taking its place in the trie as a sibling of q . We now discuss the problem of *mate selection*.

It is desirable to select mates in a manner that tends to equalize load distribution among peers. However, what constitutes “load” may depend on the particular application; in fact, peers in the same P2P network may have differing concepts of load. Several works in the literature propose schemes that distribute the stored data evenly. Yet, many peers may consider storage a cheap resource to contribute, and may be more interested in reducing the network bandwidth contributed to the P2P network. By this reasoning, a general policy for mate selection may be hard to devise. Therefore, we discuss a few possible heuristics.

Volume-balanced selection: The goal here is to equalize the volumes of peer regions, by selecting mates with probability proportional the volume of their area. This can be done by selecting a point $x \in U$ from the search space, uniformly at random, The peer q whose region contains x is designated as mate. This protocol requires $O(\log n)$ routing messages, in order to route from the bootstrap to the mate.

Data-balanced selection: If an estimate of the distribution of indexed data is available, it may be used to equalize the number of keys stored by peers. A method similar to volume-balanced selection can be used: choose some point $x \in U$ according to the estimated data distribution, and locate the corresponding peer in $O(\log n)$ messages.

Uniform selection: Random walks in a P2P network can be used to sample peers roughly uniformly [13]. In general, a walk of $O(\log n)$ hops is sufficient. In our network, each peer can have a rough estimate of $\log n$ in the following way: each message routed through the network, carries a counter with the number of hops from the originating peer. Each peer observes the counter of messages routed through it, and maintains the maximum value of these counters, which is $O(\log n)$ w.h.p. When a peer is asked to bootstrap the join of a new peer, it performs a random walk of length equal to its estimate of $\log n$. One of the peers visited during the walk is selected as mate, according to some criterion (e.g., randomly, most loaded, etc.). This protocol needs $O(\log n)$ messages.

6. APPLICATIONS OF GRASP

We now turn our attention to specific range-search applications of GRaSP, which we study experimentally in the next section.

6.1 Multidimensional Range Search

In d -dimensional range search, the search space is $[0, 1]^d$, keys are either d -dimensional points or rectangles, and ranges are d -dimensional rectangles. This type of search arises in numerous applications, and has recently received significant attention in the context of P2P networks [33, 9, 12, 28, 15, 4, 30].

For this type of problem, a natural choice for space partitioning is based on the idea of k -d trees. We can view a trie as a k -d tree, and split the space along one dimension each time, cycling through dimensions as we descend. If the expected data and query distributions are known, splits may not be even (similar to MURK [12]); typically, splits are

even. Note that, in contrast to k -d trees in main memory, we are not concerned with keeping the trie balanced.

6.2 Three-sided Range Search

A *3-sided query* is a special case of 2-d range query, where the keys are planar points and a range is a rectangle bounded from 3 sides only; a range is determined by parameters (a, b, c) , and reports all keys (x, y) with $a \leq x < b$ and $y < c$. Three-sided search arises in a large number of applications, and has been studied extensively. Optimal data structures are known both on main memory [26] and on disk [5].

To our knowledge, 3-sided search has not been studied before on P2P networks. In principle, it could be handled as a special case of 2-d range search, e.g., by the techniques of the previous section. In practice, such an approach would not be scalable, because of increased congestion. The problem lies in the shape of the queries: points with a low y -coordinate are much more likely to be returned than points with higher y -coordinates, even for uniformly distributed data and queries, inducing undue load on the peers that store them. The imbalance in access frequencies is not detrimental for data structures—rather, it can be exploited, via caching, to *improve* performance. In a P2P setting though, accesses to data stored in the network should be relatively balanced, to avoid hotspots.

We now present a solution that ameliorates this problem by employing the versatility of GRASP, to reduce contention by mapping the search problem within the setting of GRASP such that accesses are more evenly distributed. The salient feature of our solution is that it attempts to introduce storage redundancy in such a way, that frequently accessed points (those with low y -coordinates) are stored in multiple peers, so that accesses to these keys can be distributed among the copies.

6.2.1 Space-partitioning for 3-sided queries

In order to introduce redundancy within the model of GRASP, we adopt a different but equivalent definition. The search space considered is the unit square $[0, 1]^2$. Keys are defined as vertical segments, originating at some point (x, y) and extending upwards to point $(x, 1)$. Thus, point (x, y) fully determines each segment. Query ranges are defined by horizontal segments, with endpoints (a, c) and (b, c) , for $a \leq b$. Observe that this formulation is equivalent to the definition of 3-sided search given in the beginning of this section, and is also fully compatible with the GRASP model of §3.4.

Designing a good space partition in GRASP entails some broad assumptions on the distribution of data and queries to be served. For the problem at hand, our main assumption is with regard to queries. We assume that query height (the c parameter) is uniformly distributed in $[0, 1]$, and that the width of a query ($b - a$) is inversely related to query height. This assumption implies that we should handle queries of different aspect ratios, from “tall-and-narrow” to “short-and-wide” ones. These assumptions hold for many real 3-sided workloads.

We now define a space partitioning function S that caters to our assumptions. Our function is parameterized by a fixed parameter $0 < \lambda < 1$. Given the range $S(u)$ for some binary string u , we construct $S(u \cdot 0)$ and $S(u \cdot 1)$ by the following rules:

1. If u is a binary string which does not contain any 1s, then we split $S(u)$ by a horizontal line, so that the heights of the resulting parts are fixed fractions, λ and $1 - \lambda$ respectively, of the initial height. We assign the lower part to $S(u \cdot 0)$, and the upper part to $S(u \cdot 1)$.
2. Else, if u contains some (non-zero) number of 1s, we split $S(u)$ by a vertical line, into two equal regions. The left region is assigned to $S(u \cdot 0)$ and the right region to $S(u \cdot 1)$.

A possible space partition obtained by the above rules is seen in Fig. 2. It corresponds to a hypothetical network

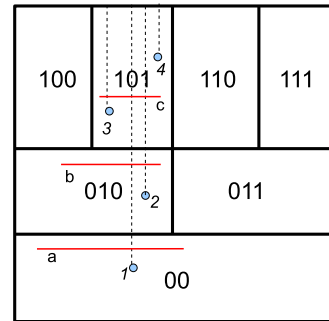


Figure 2: A 3-sided space partition for seven peers, marked by peer IDs.

of seven peers. This example also depicts four data points, marked 1 to 4, and three queries, marked a to c. Point 1 is relatively low, and is accessed by all three queries. However, point 1 is also replicated; it stored in peers 00, 010 and 101. Although all three queries shown will indeed return point 1, each query will access a different copy. For example, query a will access the copy in peer 00, whereas query c will access the copy in 101.

Our space partitioning scheme attempts to distribute the load rather than the data, by assigning peers into horizontal zones, so that each query will be answered by peers in the same zone. Peers in higher zones are assigned taller and narrower regions, whereas peers in lower zones are assigned shorter, wider regions. Thus, as long as query width decreases for taller queries, the number of peers accessed by each query will be relatively small. As this scheme redundancy, the total storage occupied in all peers may grow large, depending on the distribution of data and parameter λ . Fortunately, the redundancy is at most proportional to the number of zones, which should not grow too large, as the height of zones decreases exponentially. Also, if data is uniformly distributed, then the overall storage redundancy is constant, for any number of zones.

7. EXPERIMENTAL EVALUATION

In order to evaluate our techniques empirically, we performed extensive simulation of GRASP, for the problems of §6. The results presented below validate our analytical claims. For brevity, we refer to the technique of §6.1 as 2DRS and to the technique of §6.2 as 3SIDED.

7.1 Performance metrics

We evaluated our networks by three performance metrics. Latency is the maximum number of hops, starting

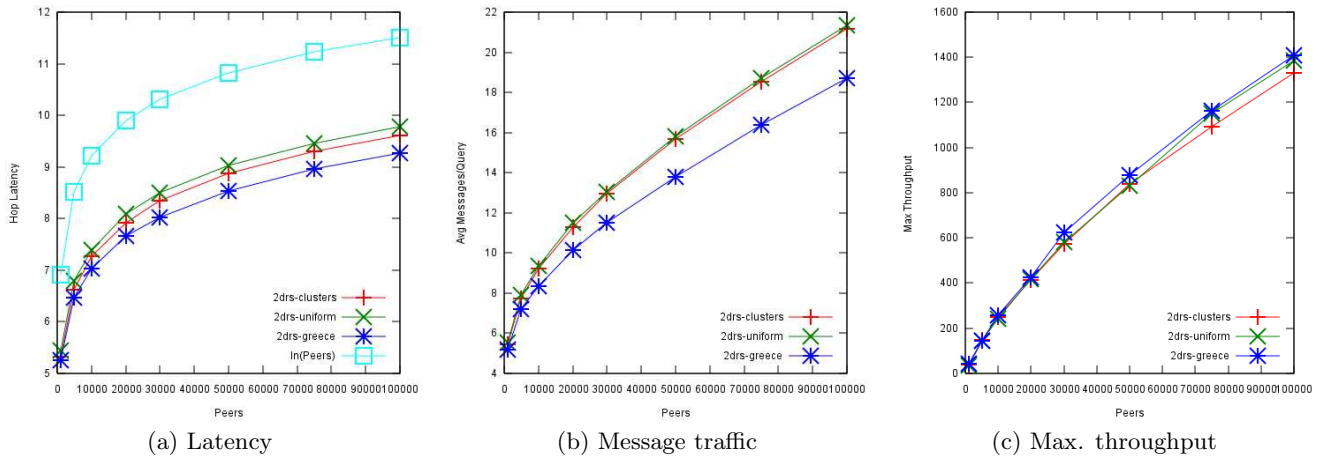


Figure 3: Performance of GRASP for 2-d range search, over network size.

from the initial peer, needed to reach any peer during the search. Message traffic M is the expected number of messages per query. Maximum throughput was proposed in [8], to quantify the resilience of a P2P network to contention by concurrent searches. Succinctly, assume that a workload of Q queries is executed on a network. For each peer p , let m_p be the number of messages received by p due to the Q queries. Assume further, that each peer can process at most one message per unit of time. Now, if queries from the workload arrive (stochastically) at a rate of Λ queries per unit of time, each query with equal probability, then messages to peer p shall arrive at a rate of $\frac{m_p}{Q}\Lambda$ messages per unit of time. Assuming that peer p is not overloaded (messages do not arrive faster than it can process them), we have $\Lambda < \frac{Q}{m_p}$. Now, maximum throughput Λ_{\max} is defined as the maximum value of Λ such that no peer is overloaded:

$$\Lambda_{\max} = \frac{Q}{\max_p m_p}$$

Also, let M be the message traffic (defined above). Then, $\Lambda_{\max} \leq n/M$, with equality holding in the ideal case where all traffic is distributed equally. Then, the ratio of traffic of the most loaded peer, over the average peer traffic, is $\frac{n/M}{\Lambda_{\max}}$.

7.2 Test workloads

In our experiments we used three datasets of 2-d points, of one million points each.

Uniform: This dataset consists of points distributed uniformly in the search space.

Clusters: This dataset contains 10 equal-size clusters, with centers uniformly located in the search space. Each cluster comprises a set of points distributed normally around its center.

Greece: This dataset was constructed from real geographic data; it contains random points along the road network of Greece.

For each of the above datasets, we constructed queries synthetically. All queries return an answer of 50 to 60 keys.

For 2DRS, a query is square, centered around a randomly chosen point from the dataset. The queries constructed thus, are distributed in the search space similarly to the dataset.

For 3SIDED, queries are constructed by selecting the midpoint of the corresponding horizontal segment uniformly at random. Thus, the position of these queries does not follow the distribution of data; however, as query size is kept bounded, the width of each query is related to data distribution (queries of the same height are narrower, if located in regions with a lot of data).

7.3 Methodology

We constructed the simulated networks by allowing peers to join the network one by one, until the desired number of peers was reached. Mate selection policies were chosen to match the indexed workload: for 2DRS networks, mate selection is data-balanced, while for 2SIDED networks, mate selection is volume-balanced.

For each type of network, we scaled the network size from 1K to 100K peers. For each network configuration (protocol, workload, size) we ran 10 simulations and averaged the results.

7.4 Results for 2DRS

The results for 2DRS are depicted in Fig. 3.

Fig. 3(a) shows that expected latency is bounded by $H_n = \ln n + O(1)$ ($\ln n$ is also plotted for reference), as Thm. 2 predicts.

The per-query message traffic M , shown in Fig. 3(b), is quite low, but grows with latency and also linearly with network size. This is only natural, since for smaller networks, the average number of keys per peer is high, and ideally queries can be covered by a single peer. However, for our largest network, a query requires, on average, at least 6 peers, to be covered. For the network of 100k peers, adding the expected latency to 6 gives a minimum of 16 messages, which is very close the value of Fig. 3(b).

For max. throughput, we note that the network scales well for all workloads, up to a value of ≈ 1400 . To put this value in context, observe that if network traffic was distributed equally among all peers, then we would have a $\Lambda_{\max} = \frac{n}{M}$ (where M is the average per-query number of messages). For $n = 100k$, the ideal Λ_{\max} would be 5000. Thus, the most loaded peer has no more than 3.5 times the average load.

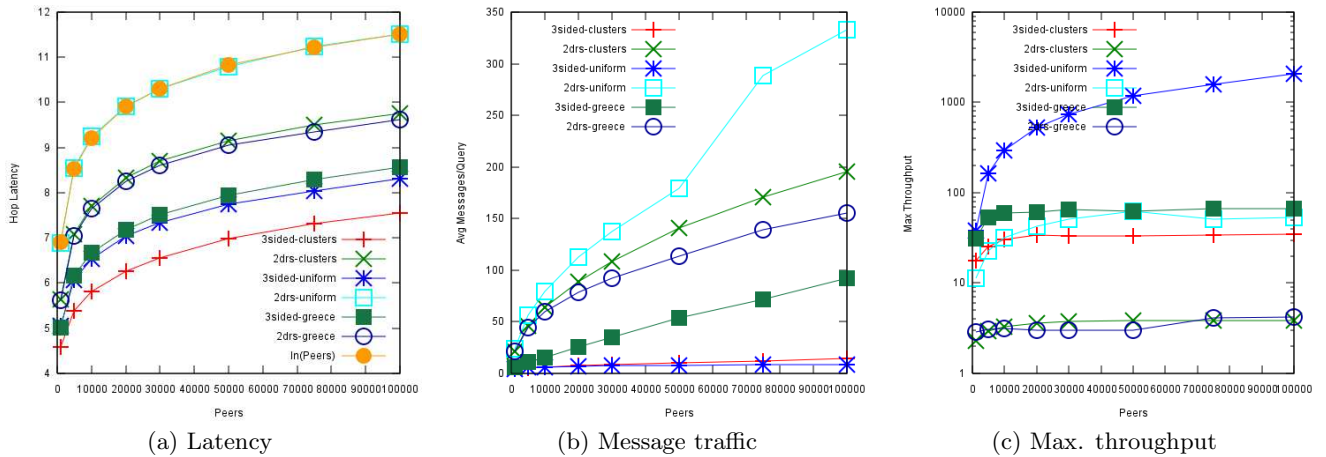


Figure 4: Performance of GRaSP for 3-sided range search, over network size.

7.5 Results for 3SIDED

The 3-sided workloads were used to evaluate both the 3SIDED protocol and the 2DRS protocol. All 3SIDED networks were constructed with a topology parameterized by $\lambda = 5/6$ (defined in §6.2).

Based on our analysis, we expect that the 2DRS protocol will exhibit good latency, but will suffer in terms of message traffic per query, and in terms of max. throughput, while the 3SIDED protocol would exploit redundancy to scale better. The results can be seen in Fig. 4.

Latency (Fig. 4(a)) is again bounded by H_n , although 3SIDED performs up to 3 hops less on average.

Message traffic (Fig. 4(b)) varies greatly between 3SIDED and 2DRS. In the worst case (2DRS over the Uniform dataset), message traffic grows roughly with \sqrt{n} ; the space partitioning of 2DRS is not well-matched to tall-and-narrow 3-sided ranges. By contrast, the message traffic for 3SIDED is much lower. In fact, for the Uniform dataset, message traffic is close to the optimal value of about 15 (as discussed in the previous section).

As predicted, Fig. 4(c) depicts huge variance in maximum throughput (note the log-scale), where 2DRS networks fail to grow above a value of 5. This is indicative of great imbalance in load distribution: even admitting a value of ≈ 300 messages per search, the ideal max. throughput is ≈ 300 . Thus, the most loaded 2DRS peer has more than 60 times the average load. For 3SIDED networks, there are mixed results. For the Uniform dataset, the number of messages is low and scalability is excellent. For the skewed datasets, the 3SIDED space partitioning is not as effective, although it is better than 2DRS by almost an order of magnitude.

We also measured storage redundancy (average number of replicas per key) incurred by the 3SIDED space partitions. Fig. 5 shows that redundancy remains bounded between 4 and 6, with the Uniform dataset obtaining the worst value of $6 = \frac{1}{1-\lambda}$. Lower values of λ would incur lower redundancy, but would also decrease maximum throughput.

The overall conclusion drawn from these results is that GRaSP exhibits excellent latency, under all circumstances. However, scalability under high throughput is less robust, decreasing significantly as the mismatch between the adopted space partition and the distribution of incoming searches

grows. However, choosing the space partition carefully, introducing redundancy by reformulating the problem appropriately, can improve scalability significantly, without additional measures.

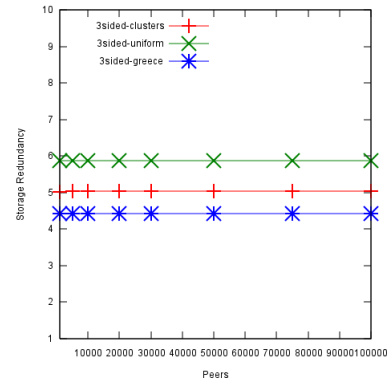


Figure 5: Storage redundancy for 3SIDED networks.

8. CONCLUSIONS

We have presented a framework for generalized range search over trie-structured P2P networks, which is motivated by new analytical results on the properties of randomized trie-based networks, such as P-Grid. We have shown that trie-based networks have logarithmic routing diameter almost always, and do not suffer from endogenous congestion, regardless of the shape or height of the underlying trie. Based on these discoveries, we developed a framework which can handle general range search problems, customized only via a space partitioning function. The salient feature of our framework is that it decouples routing concerns from the range search problem. We have evaluated our protocols on two fundamental range search problems, and we have validated our theoretical results. In particular, with our study of 3-sided range search, we have introduced a novel strategy to balance load in skewed search problems, by introducing redundancy via the appropriate choice of search space.

Our ultimate vision is towards a robust, well-engineered

peer implementation, with a focus on efficient, versatile network routing, which can be used as a black box (with minimum customization) in a number of diverse range search applications. Significant open problems remain towards our vision. A better understanding of the relationship between trie shape and load balancing is required. Also, mate selection protocols which exploit dynamically the load distribution should be considered. General dynamic load balancing (and its relationship to trie structure) is also a promising direction. Last, but certainly not least, evaluation of a real implementation is needed, as it is likely to reveal hitherto unforeseen problems and opportunities.

9. REFERENCES

- [1] ABERER, K. Scalable data access in peer-to-peer systems using unbalanced search trees. In *WDAS* (2002), pp. 107–120.
- [2] ABERER, K., CUDRÉ-MAUROUX, P., DATTA, A., DESPOTOVIC, Z., HAUSWIRTH, M., PUNCEVA, M., AND SCHMIDT, R. P-Grid: A self-organizing structured P2P system. *SIGMOD Record* 32, 3 (2003).
- [3] ABERER, K., DATTA, A., HAUSWIRTH, M., AND SCHMIDT, R. Indexing data-oriented overlay networks. In *VLDB* (2005), pp. 685–696.
- [4] ARGE, L., EPPSTEIN, D., AND GOODRICH, M. T. Skip-webs: Efficient distributed data structures for multi-dimensional data sets. In *PODC* (2005), pp. 69–76.
- [5] ARGE, L., SAMOLADAS, V., AND VITTER, J. On two-dimensional indexability and optimal range search indexing. In *PODS* (1999).
- [6] ASPNES, J., AND SHAH, G. Skip graphs. In *SODA* (2003), pp. 384–393.
- [7] BHARAMBE, A. R., AGRAWAL, M., AND SESHAN, S. Mercury: Supporting scalable multi-attribute range queries. In *SIGCOMM* (2004), pp. 353–366.
- [8] BLANAS, S., AND SAMOLADAS, V. Contention-based performance evaluation of multidimensional range search in peer-to-peer networks. In *InfoScale* (2007).
- [9] CHAWATHE, Y., RAMABHADHRAN, S., RATNASAMY, S., LAMARCA, A., SHENKER, S., AND HELLERSTEIN, J. A case study in building layered DHT applications. In *SIGCOMM* (2005), pp. 97–108.
- [10] CRAINICEANU, A., LINGA, P., GEHRKE, J., AND SHANMUGASUNDARAM, J. Querying peer-to-peer networks using P-trees. In *WebDB* (2004), pp. 25–30.
- [11] DATTA, A., HAUSWIRTH, M., JOHN, R., SCHMIDT, R., AND ABERER, K. Range queries in trie-structured overlays. In *P2P* (2005), pp. 57–66.
- [12] GANESAN, P., YANG, B., AND GARCIA-MOLINA, H. One torus to rule them all: Multidimensional queries in P2P systems. In *WebDB* (2004), pp. 19–24.
- [13] GKANTSIDIS, C., MIHAIL, M., AND SABERI, A. Random walks in peer-to-peer networks: algorithms and evaluation. *Perform. Eval.* 63, 3 (2006), 241–263.
- [14] GOODRICH, M. T., NELSON, M. J., AND SUN, J. Z. The rainbow skip graph: a fault-tolerant constant-degree distributed data structure. In *SODA* (2006), pp. 384–393.
- [15] GUPTA, A., AGRAWAL, D., AND EL ABBADI, A. Approximate range selection queries in peer-to-peer systems. In *CIDR* (2003).
- [16] HARVEY, N. J. A., JONES, M. B., SAROIU, S., THEIMER, M., AND WOLMAN, A. Skipnet: a scalable overlay network with practical locality properties. In *USENIX Symp. on Internet Technologies and Systems* (2003), pp. 9–9.
- [17] HELLERSTEIN, J., NAUGHTON, J., AND PFEFFER, A. Generalized search trees for database systems. In *Proceedings of the 21st VLDB Conference* (1995).
- [18] HUEBSCH, R., HELLERSTEIN, J. M., LANHAM, N., LOO, B. T., SHENKER, S., AND STOICA, I. Querying the internet with PIER. In *VLDB* (2003), pp. 321–332.
- [19] JAGADISH, H. V., OOI, B. C., TAN, K.-L., VU, Q. H., AND ZHANG, R. Speeding up search in peer-to-peer networks with a multi-way tree structure. In *SIGMOD* (2006), pp. 1–12.
- [20] JAGADISH, H. V., OOI, B. C., AND VU, Q. H. BATON: A balanced tree structure for peer-to-peer networks. In *VLDB* (2005), pp. 661–672.
- [21] JAGADISH, H. V., OOI, B. C., VU, Q. H., ZHANG, R., AND ZHOU, A. VBI-tree: A peer-to-peer framework for supporting multi-dimensional indexing schemes. In *ICDE* (2006), p. 34.
- [22] KALNIS, P., NG, W. S., OOI, B. C., AND TAN, K.-L. Answering similarity queries in peer-to-peer networks. *Inf. Syst.* 31, 1 (2006), 57–72.
- [23] KARGER, D. R., AND RUHL, M. Simple efficient load balancing algorithms for peer-to-peer systems. In *SPAA* (2004), pp. 36–43.
- [24] KARP, R. M. Probabilistic recurrence relations. In *STOC* (1991), pp. 190–197.
- [25] MAYMOUNKOV, P., AND MAZIÈRES, D. Kademia: A peer-to-peer information system based on the XOR metric. In *IPTPS* (2002), pp. 53–65.
- [26] MCCREIGHT, E. Priority search trees. *SIAM Journal of Computing* 14, 2 (1985), 257–276.
- [27] RATNASAMY, S., FRANCIS, P., HANDLEY, M., KARP, R., AND SCHENKER, S. A scalable content-addressable network. In *SIGCOMM* (2001), pp. 161–172.
- [28] RATNASAMY, S., KARP, B., LI, Y., YU, F., ESTRIN, D., GOVINDAN, R., AND SHENKER, S. GHT: A geographic hash table for data-centric storage. In *WSNA* (2002), pp. 78–87.
- [29] ROWSTRON, A., AND DRUSCHEL, P. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *MIDDLEWARE* (2001), pp. 329–350.
- [30] SHU, Y., OOI, B. C., TAN, K.-L., AND ZHOU, A. Supporting multi-dimensional range queries in peer-to-peer systems. In *P2P* (2005), pp. 173–180.
- [31] STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, M. F., AND BALAKRISHNAN, H. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM* (2001), pp. 149–160.
- [32] TANG, C., DWARKADAS, S., AND XU, Z. On scaling latent semantic indexing for large peer-to-peer systems. In *SIGIR* (2004), pp. 112–121.
- [33] ZHENG, C., SHEN, G., LI, S., AND SHENKER, S. Distributed segment tree: Support of range query and cover query over DHT. In *IPTPS* (2006).