*** Article**

Survey of Research towards Robust Peer-to-Peer Networks: Search Methods

John Risson [a,*], Tim Moors [a]

[a] *School of Electrical Engineering and Telecommunications, University of New South Wales, Sydney, New South Wales, 2052, Australia.*

Abstract

The pace of research on peer-to-peer (P2P) networking in the last five years warrants a critical survey. P2P has the makings of a disruptive technology - it can aggregate enormous storage and processing resources while minimizing entry and scaling costs. Failures are common amongst massive numbers of distributed peers, though the impact of individual failures may be less than in conventional architectures. Thus the key to realizing P2P's potential in applications other than casual file sharing is robustness.

P2P search methods are first couched within an overall P2P taxonomy. P2P indexes for simple key lookup are assessed, including those based on Plaxton trees, rings, tori, butterflies, de Bruijn graphs and skip graphs. Similarly, P2P indexes for keyword lookup, information retrieval and data management are explored. Finally, early efforts to optimize range, multi-attribute, join and aggregation queries over P2P indexes are reviewed. Insofar as they are available in the primary literature, robustness mechanisms and metrics are highlighted throughout. However, the low-level mechanisms that most affect robustness are not well isolated in the literature. Furthermore, there has been little consensus on robustness metrics. Recommendations are given for future research.

***Keywords**:* Peer-to-peer network; Distributed hash table; Consistent hashing; Scalable distributed data structure; Vector model; Latent semantic indexing; Dependability; Torus; Butterfly network; Skip graph; Plaxton tree, de Bruijn graph

[*] Corresponding author. Tel.: +61 410 285 215. Fax: +61 2 9385 5993.
E-mail addresses: jr@tuffit.com (John Risson), t.moors@unsw.edu.au (Tim Moors)

1. Introduction

Peer-to-peer (P2P) networks are those that exhibit three characteristics: self-organization, symmetric communication and distributed control [1]. A self-organizing P2P network "automatically adapts to the arrival, departure and failure of nodes" [2]. Communication is symmetric in that peers act as both clients and servers. It has no centralized directory or control point. USENET servers or BGP peers have these traits [3] but the emphasis here is on the flurry of research since 2000. Leading examples include Gnutella [4], Freenet [5], Pastry [2], Tapestry [6], Chord [7], the Content Addressable Network (CAN) [8], pSearch [9] and Edutella [10]. Some have suggested that peers are inherently unreliable [11]. Others have assumed well-connected, stable peers [12].

This critical survey of P2P academic literature is warranted, given the intensity of recent research. At the time of writing, one research database lists over 5,800 P2P publications [13]. One vendor surveyed P2P products and deployments [14]. There is also a tutorial survey of leading P2P systems [15]. DePaoli and Mariani recently reviewed the dependability of some early P2P systems at a high level [16]. The need for a critical survey was flagged in the peer-to-peer research group of the Internet Research Task Force (IRTF) [17].

P2P is potentially a disruptive technology with numerous applications, but this potential will not be realized unless it is demonstrated to be robust. A massively distributed search technique may yield numerous practical benefits for applications [18]. A P2P system has potential to be more dependable than architectures relying on a small number of centralized servers. It has potential to evolve better from small configurations - the capital outlays for high performance servers can be reduced and spread over time if a P2P assembly of general purpose nodes is used. A similar argument motivated the deployment of distributed databases – one thousand, off-the-shelf PC processors are more powerful and much less expensive than a large mainframe computer [19]. Storage and processing can be aggregated to achieve massive scale. Wasteful partitioning between servers or clusters can be avoided. As Gedik and Liu put it, if P2P is to find its way into applications other than casual file sharing, then reliability needs to be addressed [20].

The taxonomy of Figure 1 divides the entire body of P2P research literature along four lines: search, storage, security and applications. This survey concentrates on search aspects. A P2P search network consists of an underlying index (Sections 2 to 4) and queries that propagate over that index (Section 5).

This survey is concerned with two questions. The first is "How do P2P search networks work?" This foundation is important given the pace and breadth of P2P research in the last five years. In Section 2, we classify indexes as local, centralized and distributed. Since distributed indexes are becoming dominant, they are given closer attention in Sections 3 and 4. Section 3 gives a two-tiered comparison of distributed P2P indexes for simple key lookup. The top tier explores their overall origins (Section 3.1) and robustness (Section 3.2). The second tier (Sections 3.3 to 3.8) divides them by index structure, in particular Plaxton trees, rings, tori, butterflies, de Bruijn graphs and skip graphs. Section 4 reviews distributed P2P indexes supporting keyword lookup (Section 4.1) and information retrieval (Section 4.2). Section 5 probes the embryonic research on P2P queries, in particular, range queries (Section 5.1), multi-attribute queries (Section 5.2), join queries (Section 5.3) and aggregation queries (Section 5.4).

The second question is "How robust are P2P search networks?" Insofar as it is available in the research literature, we tease out the robustness mechanisms and metrics throughout Sections 2 to 5. Unfortunately, robustness is often more sensitive to low-level design choices than it is to the broad P2P index structure, yet these underlying design choices are seldom isolated in the primary literature [229]. Furthermore, there has been little consensus on P2P robustness metrics (Section 3.2). Section 6 gives recommendations to address these important gaps.

1.1. Related Disciplines

Peer-to-peer research draws upon numerous distributed systems disciplines. Networking researchers will recognize familiar issues of naming, routing and congestion control. P2P designs need to address routing and security issues across network region boundaries [152]. Networking research has traditionally been host-centric. The web's Universal Resource Identifiers are naturally tied to specific hosts, making object mobility a challenge [216].

P2P work is data-centric [230]. P2P systems for dynamic object location and routing have borrowed heavily from the distributed systems corpus. Some have used replication, erasure codes and Byzantine agreement [111]. Others have used epidemics for durable peer group communication [39].

Similarly, P2P research is set to benefit from database research [231]. Database researchers will recognize the need to reapply Codd's principle of physical data independence, that is, to decouple data indexes from the applications that use the data [23]. It was the invention of appropriate indexing mechanisms and query optimizations

that enabled data independence. Database indexes like B+ trees have an analog in P2P's distributed hash tables (DHTs). Wide-area, P2P query optimization is a ripe, but challenging, area for innovation.

More flexible distribution of objects comes with increased security risks. There are opportunities for security researchers to deliver new methods for availability, file authenticity, anonymity and access control [25]. Proactive and reactive mechanisms are needed to deal with large numbers of autonomous, distributed peers. To build robust systems from cooperating but self-interested peers, issues of identity, reputation, trust and incentives need to be tackled. Although it is beyond the scope of this paper, robustness against malicious attacks also ought to be addressed [195].

Possibly the largest portion of P2P research has majored on basic routing structures [18], where research on algorithms comes to the fore. Should the overlay be "structured" or "unstructured"? Are the two approaches competing or complementary? Comparisons of the "structured" approaches – hypercubes, rings, toroids, butterflies, de Bruijn and skip graphs – have weighed the amount of routing state per peer and the number of links per peer against overlay hop-counts. While "unstructured" overlays initially used blind flooding and random walks, overheads usually trigger some structure, for example super-peers and clusters.

P2P applications rely on cooperation between these disciplines. Applications have included file sharing, directories, content delivery networks, email, distributed computation, publish-subscribe middleware, multicasting, and distributed authentication. Which applications will be suited to which structures? Are there adaptable mechanisms which can decouple applications from the underlying data structures? What are the criteria for selection of applications amenable to a P2P design [1]?

Robustness is emphasized throughout the survey. We are particularly interested in two aspects. The first, dependability, was a leading design goal for the original Internet [232]. It deserves the same status in P2P. The measures of dependability are well established: reliability, a measure of the mean-time-to-failure (MTTF); availability, a measure of both the MTTF and the mean-time-to-repair (MTTR)[1]; maintainability; and safety [233]. The second aspect is the ability to accommodate variation in outcome, which one could call adaptability. Its measures have yet to be defined. In the context of the Internet, it was only recently acknowledged as a first class

[1] Traditionally, availability = MTTF / (MTTF + MTTR)

requirement [234]. In P2P, it means planning for the tussles over resources and identity. It means handling different kinds of queries and accomodating changeable application requirements with minimal intervention. It means "organic scaling" [22], whereby the system grows gracefully, without a priori data center costs or architectural breakpoints.

In the following section, we discuss one notable omission from the taxonomy of P2P networking in Figure 1 - routing.

1.2. Structured and Unstructured Routing

P2P routing algorithms have been classified as "structured" or "unstructured". Early instantiations of Gnutella were unstructured – keyword queries were flooded widely [235]. Napster [236] had decentralized content and a centralized index, so only partially satisfies the distributed control criteria for P2P systems. Early structured algorithms included Plaxton, Rajaraman and Richa (PRR) [30], Pastry [2], Tapestry [31], Chord [7] and the Content Addressable Network [8]. Mishchke and Stiller recently classified P2P systems by the presence or absence of structure in routing tables and network topology [237].

Some have cast unstructured and structured algorithms as competing alternatives. Unstructured approaches have been called "first generation", implicitly inferior to the "second generation" structured algorithms [2, 31]. When generic key lookups are required, these structured, key-based routing schemes can guarantee location of a target within a bounded number of hops [23]. The broadcasting unstructured approaches, however, may have large routing costs, or fail to find available content [22]. Despite the apparent advantages of structured P2P, several research groups are still pursuing unstructured P2P.

There have been two main criticisms of structured systems [61]. The first relates to peer transience, which in turn affects robustness. Chawathe et al. opined that highly transient peers are not well supported by DHTs [61]. P2P systems often exhibit "churn", with peers continually arriving and departing. One objection to concerns about highly transient peers is that many applications use peers in well-connected parts of the network. The Tapestry authors analysed the impact of churn in a network of 1000 nodes [31]. Others opined that it is possible to maintain a robust DHT at relatively low cost [238]. Very few papers have quantitatively *compared* the resilience of structured systems. Loguinov, Kumar et al claimed that there were only two such works [24, 36].

The second criticism of structured systems is that they do not support keyword searches and complex queries as well as unstructured systems. Given the current file-sharing deployments, keyword searches seem more important than exact-match key searches in the short term. Paraphrased, “most queries are for hay, not needles” [61].

More recently, some have justifiably seen unstructured and structured proposals as complementary, not competing [239]. Their starting point was the observation that unstructured flooding or random walks are inefficient for data that is not highly replicated across the P2P network. Structured graphs can find keys efficiently, irrespective of replication. Castro et al proposed Structella, a hybrid of Gnutella built on top of Pastry [239]. Another design used structured search for rare items and unstructured search for massively replicated items [54].

However, the “structured versus unstructured routing” taxonomy is becoming less useful, for two reasons, Firstly, most “unstructured” proposals have evolved and incorporated structure. Consider the classic “unstructured” system, Gnutella [4]. For scalability, its peers are either ultrapeers or leaf nodes. This hierarchy is augmented with a query routing protocol whereby ultrapeers receive a hashed summary of the resource names available at leaf-nodes. Between ultrapeers, simple query broadcast is still used, though methods to reduce the query load here have been considered [240]. Secondly, there are emerging schema-based P2P designs [59], with super-node hierarchies and structure within documents. These are quite distinct from the structured DHT proposals.

Given that most, if not all, P2P designs today assume some structure, a more instructive taxonomy would describe the structure. In this survey, we use a database taxonomy in lieu of the networking taxonomy, as suggested by Hellerstein, Cooper and Garcia-Molina [23, 241]. The structure is determined by the type of index. Queries feature in lieu of routing. The DHT algorithms implement a “semantic-free index” [216]. They are oblivious of whether keys represent document titles, meta-data, or text. Gnutella-like and schema-based proposals have a “semantic index”.

1.3. Indexing and Searching

Index engineering is at the heart of P2P search methods. It captures a broad range of P2P issues, as demonstrated by the Search/Index Links model [241]. As Manber put it, “the most important of the tools for information retrieval is the index—a collection of terms with pointers to places where information about documents can be found”[242]. Sen and Wang noted that a “P2P network” usually consists of connections between hosts for application-layer

signaling, rather than for the data transfer itself [243]. Similarly, we concentrate on the "signaled" indexes and queries.

Our focus here is the dependability and adaptability of the search network. Static dependability is a measure of how well queries route around failures in a network that is normally fault-free. Dynamic dependability gives an indication of query success when nodes and data are continually joining and leaving the P2P system. An adaptable index accommodates change in the data and query distribution. It enables data independence, in that it facilitates changes to the data layout without requiring changes to the applications that use the data [23]. An adaptable P2P system can support rich queries for a wide range of applications. Some applications benefit from simple, semantic-free key lookups [244]. Others require more complex, Structured Query Language (SQL)-like queries to find documents with multiple keywords, or to aggregate or join query results from distributed relations [22].

2. Index Types

A P2P index can be *local*, *centralized* or *distributed*. With a local index, a peer only keeps the references to its own data, and does not receive references for data at other nodes. The very early Gnutella design epitomized the local index (Section 2.1). In a centralized index, a single server keeps references to data on many peers. The classic example is Napster (Section 2.2). With distributed indexes, pointers towards the target reside at several nodes. One very early example is Freenet (Section 2.3). Distributed indexes are used in most P2P designs nowadays – they dominate this survey.

P2P indexes can also be classified as *non-forwarding* and *forwarding*. When queries are guided by a non-forwarding index, they jump to the node containing the target data in a single hop. There have been semantic and semantic-free one-hop schemes [138, 245, 246]. Where scalability to a massive number of peers is required, these schemes have been extended to two-hops [247, 248]. More common are the forwarding P2Ps where the number of hops varies with the total number of peers, often logarithmically. The related tradeoffs between routing state, lookup latency, update bandwidth and peer churn are critical to total system dependability.

2.1. Local Index

P2Ps with a purely local data index are becoming rare. In such designs, peers flood queries widely and only index their own content. They enable rich queries – the search is not limited to a simple key lookup. However, they also generate a large volume of query traffic with no guarantee that a match will be found, even if it does exist on the network. For example, to find potential peers on the early instantiations of Gnutella, 'ping' messages were broadcast over the P2P network and the 'pong' responses were used to build the node index. Then small 'query' messages, each with a list of keywords, are broadcast to peers which respond with matching filenames [4].

There have been numerous attempts to improve the scalability of local-index P2P networks. Gnutella uses fixed time-to-live (TTL) rings, where the query's TTL is set less than 7-10 hops [4]. Small TTLs reduce the network traffic and the load on peers, but also reduce the chances of a successful query hit. One paper reported, perhaps a little too bluntly, that the fixed "TTL-based mechanism does not work" [67] To address this TTL selection problem, they proposed an expanding ring, known elsewhere as iterative deepening [29]. It uses successively larger TTL counters until there is a match. The flooding, ring and expanding ring methods all increase network load with duplicated query messages. A random walk, whereby an unduplicated query wanders about the network, does indeed reduce the network load but massively increases the search latency. One solution is to replicate the query k times at each peer. Called random k-walkers, this technique can be coupled with TTL limits, or periodic checks with the query originator, to cap the query load [67]. Adamic, Lukose et al. suggested that the random walk searches be directed to nodes with higher degree, that is, with larger numbers of inter-peer connections [249]. They assumed that higher-degree peers are also capable of higher query throughputs. However without some balancing design rule, such peers would be swamped with the entire P2P signaling traffic. In addition to the above approaches, there is the 'directed breadth-first' algorithm [29]. It forwards queries within a subset of peers selected according to heuristics on previous performance, like the number of successful query results. Another algorithm, called probabilistic flooding, has been modeled using percolation theory [250].

Several measurement studies have investigated locally indexed P2Ps. Jovanovic noted Gnutella's power law behaviour [70]. Sen and Wang compared the performance of Gnutella, Fasttrack [251] and Direct Connect [243, 252, 253][2]. At the time, only Gnutella used local data indexes. All three schemes now use distributed data indexes,

[2] Bearshare and Limewire clients use Gnutella. KaZaa and Grokster clients use FastTrack. When Sen and Wang wrote their 2002 paper, Morpheus also used FastTrack.

with hierarchy in the form of Ultrapeers (Gnutella), Super-Nodes (FastTrack) and Hubs (Direct Connect). It was found that a very small percentage of peers have a very high degree and that the total system dependability is at the mercy of such peers. While peer up-time and bandwidth were heavy-tailed, they did not fit well with the Zipf distribution. Fortunately for Internet Service Providers, measures aggregated by IP prefix and Autonomous System (AS) were more stable than for individual IP addresses. A study of University of Washington traffic found that Gnutella and Kazaa together contributed 43% of the university's total TCP traffic [254]. They also reported a heavy-tailed distribution, with 600 external peers (out of 281,026) delivering 26% of Kazaa bytes to internal peers. Furthermore, objects retrieved from the P2P network were typically three orders of magnitude larger than web objects – 300 objects contributed to almost half of the total outbound Kazaa bandwidth. Others reported Gnutella's topology mismatch, whereby only 2-5% of P2P connections link peers in the same AS, despite over 40% of peers being in the top 10 ASes [65]. Together these studies underscore the significance of multimedia sharing applications. They motivate interesting caching and locality solutions to the topology mismatch problem.

These same studies bear out one main dependability lesson: total system dependability may be sensitive to the dependability of high degree peers. The designers of Scamp translated this observation to the design heuristic, "have the degree of each node be of nearly equal size" [153]. They analyzed a system of N peers, with mean degree c.log(N), where link failures occur independently with probability ℳ. If $\delta>0$ is fixed and $c>(1+\delta)/(-\log(\varepsilon))$ then the probability of graph disconnection goes to zero as $N\to\infty$. Otherwise, if $c<(1-\delta)/(-\log(\varepsilon))$ then the probability of disconnection goes to one as $N\to\infty$. They presented a localizer, which finds approximate minima to a global function of peer degree and arbitrary link costs using only local information. The Scamp overlay construction algorithms could support any of the flooding and walking routing schemes above, or other epidemic and multicasting schemes for that matter. Resilience to high churn rates was identified for future study.

2.2. Central Index

Centralized schemes like Napster [236] are significant because they were the first to demonstrate the P2P scalability that comes from separating the data index from the data itself. Ultimately 36 million Napster users lost their service not because of technical failure, but because the single administration was vulnerable to the legal challenges of record companies [255].

There has since been little research on P2P systems with central data indexes. Such systems have also been called 'hybrid' since the index is centralized but the data is distributed. Yang and Garcia-Molina devised a four-way classification of hybrid systems [256]: *unchained* servers, where users whose index is on one server do not see other servers' indexes; *chained* servers, where the server that receives a query forwards it to a list of servers if it does not own the index itself; *full replication*, where all centralized servers keep a complete index of all available metadata; and *hashing*, where keywords are hashed to the server where the associated inverted list is kept. The unchained architecture was used by Napster, but it has the disadvantage that users do not see all indexed data in the system. Strictly speaking, the other three options illustrate the distributed data index, not the central index. The chained architecture was recommended as the optimum for the music-swapping application at the time. The methods by which clients update the central index were classified as *batch* or *incremental*, with the optimum determined by the query-to-login ratio. Measurements were derived from a clone of Napster called OpenNap[257]. Another study of live Napster data reported wide variation in the availability of peers, a general unwillingness to share files (20-40% of peers share few or no files), and a common understatement of available bandwidth so as to discourage other peers from sharing one's link [202].

Influenced by Napster's early demise, the P2P research community may have prematurely turned its back on centralized architectures. Chawathe, Ratnasamy et al. opined that Google and Yahoo demonstrate the viability of a centralized index. They argued that "the real barriers to Napster-like designs are not technical but legal and financial" [61]. Even this view may be a little too harsh on the centralized architectures – it implies that they always have an upfront capital hurdle that is steeper than for distributed architectures. The closer one looks at scalable 'centralized' architectures, the less the distinction with 'distributed' architectures seems to matter. For example, it is clear that Google's designers consider Google a distributed, not centralized, file system [258]. Google demonstrates the scale and performance possible on commodity hardware, but still has a centralized master that is critical to the operation of each Google cluster. Time may prove that the value of emerging P2P networks, regardless of the centralized-versus-distributed classification, is that they smooth the capital outlays and remove the single points of failure across the spectra of scale and geographic distribution.

2.3. Distributed Index

An important early P2P proposal for a distributed index was Freenet [5, 71, 259]. While its primary emphasis was the anonymity of peers, it did introduce a novel indexing scheme. Files are identified by low-level "content-hash" keys and by "secure signed-subspace" keys which ensure that only a file owner can write to a file while anyone can

read from it. To find a file, the requesting peer first checks its local table for the node with keys closest to the target. When that node receives the query, it too checks for either a match or another node with keys close to the target. Eventually, the query either finds the target or exceeds time-to-live (TTL) limits. The query response traverses the successful query path in reverse, depositing a new routing table entry (the requested key and the data holder) at each peer. The insert message similarly steps towards the target node, updating routing table entries as it goes, and finally stores the file there. Whereas early versions of Gnutella used breadth-first flooding, Freenet uses a more economic depth-first search [260].

An initial assessment has been done of Freenet's robustness. It was shown that in a network of 1000 nodes, the median query path length stayed under 20 hops for a failure of 30% of nodes. While the Freenet designers considered this as evidence that the system is "surprisingly robust against quite large failures" [71], the same datapoint may well be outside meaningful operating bounds. How many applications are useful when the first quartile of queries have path lengths of several hundred hops in a network of only 1000 nodes, per Figure 4 of [71]? To date, there has been no analysis of Freenet's dynamic robustness. For example, how does it perform when nodes are continually arriving and departing?

There have been both criticisms and extensions of the early Freenet work. Gnutella proponents acknowledged the merit in Freenet's avoidance of query broadcasting [261]. However, they are critical on two counts: the exact file name is needed to construct a query; and exactly one match is returned for each query. P2P designs using DHTs, per Section 3, share similar characteristics – a precise query yields a precise response. The similarity is not surprising since Freenet also uses a hash function to generate keys. However, the query routing used in the DHTs has firmer theoretical foundations. Another difference with DHTs is that Freenet will take time, when a new node joins the network, to build an index that facilitates efficient query routing. By the inventor's own admission, this is damaging for a user's first impressions [262]. It was proposed to download a copy of routing tables from seed nodes at startup, even though the new node might be far from the seed node. Freenet's slow startup motivated Mache, Gilbert et al. to amend the overlay after failed requests and to place additional index entries on successful requests – they claim almost an order of magnitude reduction in average query path length [260]. Clarke also highlighted the lack of locality or bandwidth information available for efficient query routing decisions [262]. He proposed that each node gather response times, connection times and proportion of successful requests for each entry in the query routing table. When searching for a key that is not in its own routing table, it was proposed to estimate response times from

the routing metrics for the nearest known keys and consequently choose the node that can retrieve the data fastest. The response time heuristic assumed that nodes close in the key space have similar response times. This assumption stemmed from early deployment observations that Freenet peers seemed to specialize in parts of the keyspace – it has not been justified analytically. Kronfol drew attention to Freenet's inability to do keyword searches [263]. He suggested that peers cache lists of weighted keywords in order to route queries to documents, using Term Frequency Inverse Document Frequency (TFIDF) measures and inverted indexes (Section 4.2.1). With these methods, a peer can route queries for simple keyword lists or more complicated conjunctions and disjunctions of keywords. Robustness analysis and simulation of Kronfol's proposal remains open.

The vast majority of P2P proposals in following sections rely on a distributed index.

3. Semantic-Free Index

Many of today's distributed network indexes are *semantic*. The semantic index is human-readable. For example, it might associate information with other keywords, a document, a database key or even an administrative domain. It makes it easy to associate objects with particular network providers, companies or organizations, as evidenced in the Domain Name System (DNS). However, it can also trigger legal tussles and frustrate content replication and migration [216].

Distributed Hash Tables (DHTs) have been proposed to provide *semantic-free*, data-centric references. DHTs enable one to find an object's persistent key in a very large, changing set of hosts. They are typically designed for [23]:

a) low degree. If each node keeps only a small number of transport connections to other nodes, the impact of high node arrival and departure rates is contained;

b) low diameter. The hops and delay introduced by the extra indirection are minimized;

c) greedy routing. Nodes independently calculate a short path to the target. At each hop, the query moves closer to the target; and

d) robustness. A path to the target can be found even when links or nodes fail.

3.1. Origins

To understand the origins of recent DHTs, one needs to look to three contributions from the 1990s. The first two - Plaxton, Rajaraman, and Richa (PRR) [30] and Consistent Hashing [49] - were published within one month of each

other. The third, the Scalable Distributed Data Structure (SDDS) [52], was curiously ignored in significant structured P2P designs despite having some similar goals [2, 6, 7]. It has been briefly referenced in other P2P papers [46, 264-267].

PRR is the most recent of the three. It influenced the designs of Pastry [2], Tapestry [6] and Chord [7]. The value of PRR is that it can locate objects using fixed-length routing tables [6]. Objects and nodes are assigned a semantic-free address, for example a 160 bit key. Every node is effectively the root of a spanning tree. A message routes toward an object by matching longer address suffixes, until it encounters either the object's root node or another node with a 'nearby' copy. It can route around link and node failure by matching nodes with a related suffix. The scheme has several disadvantages [6]: global knowledge is needed to construct the overlay; an object's root node is a single point of failure; nodes cannot be inserted and deleted; there is no mechanism for queries to avoid congestion hot spots.

Karger et al. introduced Consistent Hashing in the context of the web caching problem [49]. Web servers could conceivably use standard hashing to place objects across a network of caches. Clients could use the approach to find the objects. For normal hashing, most object references would be moved when caches are added or deleted. On the other hand, Consistent Hashing is "smooth" – when caches are added or deleted, the minimum number of object references move so as to maintain load balancing. Consistent Hashing also ensures that the total number of caches responsible for a particular object is limited. Whereas Litwin's Linear Hashing (LH*) scheme requires 'buckets' to be added one at a time in sequence [50], Consistent Hashing allows them to be added in any order [49]. There is an open Consistent Hashing problem pertaining to the fraction of items moved when a node is inserted [165]. Extended Consistent Hashing was recently proposed to randomize queries over the spread of caches to significantly reduce the load variance [268]. Interestingly, Karger [49] referred to an older DHT algorithm by Devine that used "a novel autonomous location discovery algorithm that learns the buckets' locations instead of using a centralized directory" [51].

In turn, Devine's primary point of reference was Litwin's work[3] on SDDSs and the associated LH* algorithm [52]. An SDDS satisfies three design requirements: files grow to new servers only when existing servers are well loaded; there is no centralized directory; the basic operations like insert, search and split never require atomic

[3] Both Litwin and Devine were at UC-Berkeley in 1993.

updates to multiple clients. Honicky and Miller suggested the first requirement could be considered a limitation since expansion to new servers is not under administrative control [266]. Litwin recently noted numerous similarities and differences between LH* and Chord [269]. He found that both implement key search. Although LH* refers to clients and servers, nodes can operate as peers in both. Chord 'splits' nodes when a new node is inserted, while LH* schedules 'splits' to avoid overload. Chord requests travel O(logN) hops, while LH* client requests need at most two hops to find the target. Chord stores a small number of 'fingers' at each node. LH* servers store N/2 to N addresses while LH* clients store 1 to N addresses. This tradeoff between hop count and the size of the index affects system robustness, and bears striking similarity to recent one- and two-hop P2P schemes in Section 2. The arrival and departure of LH* clients does not disrupt LH* server metadata at all. Given the size of the index, the arrival and departure of LH* servers is likely to cause more churn than that of Chord nodes. Unlike Chord, LH* has a single point of failure, the split coordinator. It can be replicated. Alternatively it can be removed in later LH* variants, though details have not been progressed for lack of practical need [269].

3.2. Dependability Comparisons

Before launching into a critique of the various DHT geometries (Sections 3.3 to 3.8), we first make four overall observations about their dependability. Dependability metrics fall into two categories: *static dependability*, a measure of performance before recovery mechanisms take over; and *dynamic dependability*, for the most likely case in massive networks where there is continual failure and recovery ("churn").

Observation A: Static dependability comparisons show that no O(log N) DHT geometry is significantly more dependable than the other O(log N) geometries. Gummadi et al. compared the tree, hypercube, butterfly, ring, XOR and hybrid geometries. In such geometries, nodes generally know about O(log N) neighbors and route to a destination in O(log N) hops, where N is the number of nodes in the overlay. Gummadi et al. asked "Why not the ring?". They concluded that only the ring and XOR geometries permit flexible choice of both neighbors and alternative routes [24]. Loguinov et al. added the de Bruijn graph to their comparison [36]. They concluded that the classical analyses, for example the probability that a particular node becomes disconnected, yield no major differences between the resilience of Chord, CAN and de Bruijn graphs. Using bisection width (the minimum edge count between two equal partitions) and path overlap (the likelihood that backup paths will encounter the same failed nodes or links as the primary path), they argued for the superior resilience of the de Bruijn graph. In short, ring, XOR and de Bruijn graphs all permit flexible choice of alternative paths, but only in de Bruijn are the alternate paths independent of each other [36].

Observation B: Dynamic dependability comparisons show that DHT dependability is sensitive to the underlying topology maintenance algorithms. Li et al. give the best comparison to date of several leading DHTs during churn [270]. They relate the disparate configuration parameters of Tapestry, Chord, Kademlia, Kelips and OneHop to fundamental design choices. For each of these DHTs, they plotted the optimal performance in terms of lookup latency (milliseconds) and fraction of failed lookups. The results led to several important insights about the underlying algorithms, for example: increasing routing table size is more cost-effective than increasing the rate of periodic stabilization; learning about new nodes during the lookup process sometimes eliminates the need for stabilization; parallel lookups reduce latency due to timeouts more effectively than faster stabilization. Similarly, Zhuang et al. compared keep-alive algorithms for DHT failure detection [271]. Such algorithmic comparisons can significantly improve the dependability of DHT designs.

In Figure 2, we propose a taxonomy for the topology maintenance algorithms that influence dependability. The algorithms can be classified by how nodes join and leave, how they first detect failures, how they share information about topology updates, and how they react when they receive information about topology updates.

Observation C: Most DHTs use O(log N) geometries to suit ephemeral nodes. The O(1) hop DHTs suit stable nodes and deserve more research attention. Most of the DHTs in Sections 3.3 to 3.8 assume that nodes are ephemeral, with expected lifetimes of one to two hours. They therefore mostly use an O(log N) geometry. The common assumption is that maintenance of full routing tables in the O(1) hop DHTs will consume excessive bandwidth when nodes are continually joining and leaving. The corollary is that, when they run on stable infrastructure servers [277], most of the DHTs in Sections 3.3 to 3.8 are less than optimal - lookups take many more hops than necessary, wasting latency and bandwidth budgets. The O(1) hop DHTs suit stable deployments and high lookup rates. For a churning 1024-node network, Li et al. concluded that OneHop is superior to Chord, Tapestry, Kademlia and Kelips in terms of latency and lookup success rate [270]. For a 3000-node network, they concluded that “OneHop is only preferable to Chord when the deployment scenario allows a communication cost greater than 20 bytes per node per second” [270]. This apparent limitation needs to be put in context. They assumed that each node issues only one lookup every 10 minutes and has a lifetime of only 60 minutes. It seems reasonable to expect that in some deployments, nodes will have a lifetime of weeks or more, a maintenance bandwidth of tens of kilobits per second, and a load of hundreds of lookups per second. O(1) hop DHTs are superior in such situations. OneHop can scale at least to many tens of thousands of nodes [247]. The recent O(1) hop designs [247, 274] are vastly

outnumbered by the O(log N) DHTs in Sections 3.3 to 3.8. Research on the algorithms of Figure 2 will also yield improvements in the dependability of the O(1) hop DHTs.

Observation D: Although not yet a mature science, the study of DHT dependability is helped by recent simulation and formal development tools. While there are recent reference architectures [273, 277], much of the DHT literature in Sections 3.3 to 3.8 does not lend itself to repeatable, comparative studies. The best comparative work to date [270] relies on the P2PSIM simulator [278]. At the time of writing, it supports more DHT geometries than any other simulator. As the study of DHTs matures, we can expect to see the simulation emphasis shift from geometric comparison to a comparison of the algorithms of Figure 2.

P2P correctness proofs generally rely on less than complete formal specifications of system invariants and events [7, 45, 279]. Li and Plaxton expressed concern that "when many joins and leaves happen concurrently, it is not clear whether the neighbor tables will remain in a 'good' state" [47]. While acknowledging that guaranteeing consistency in a failure prone network is impossible, Lynch, Malkhi et al. sketched amendments to the Chord algorithm to guarantee atomicity [280]. More recently, Gilbert, Lynch et al. gave a new algorithm for atomic read/write memory in a churning distributed network, suggesting it to be a good match for P2P [281]. Lynch and Stoica show in an enhancement to Chord that lookups are provably correct when there is a limited rate of joins and failures [282]. Fault Tolerant Active Rings is a protocol for active joins and leaves that was formally specified and proven using B-method tools [283]. A good starting point for a formal DHT development would be the numerous informal API specifications [22, 284, 285]. Such work could be informed by other efforts to formally specify routing invariants [286, 287].

In Sections 3.3 to 3.8, we introduce the main geometries for simple key lookup and survey their robustness mechanisms.

3.3. Plaxton Trees

Work began in March 2000 on a structured, fault-tolerant, wide-area Dynamic Object Location and Routing (DOLR) system called Tapestry [6, 155]. While DHTs fix replica locations, a DOLR API enables applications to control object placement [31]. Tapestry's basic location and routing scheme follows Plaxton, Rajaraman and Richa (PRR) [30], but it remedies PRR's robustness shortcomings described in Section 3.1. Whereas each object has one root node in PRR, Tapestry uses several to avoid a single point of failure. Unlike PRR, it allows nodes to be inserted

and deleted. Whereas PRR required a total ordering of nodes, Tapestry uses 'surrogate routing' to *incrementally* choose root nodes. The PRR algorithm does not address congestion, but Tapestry can put object copies close to nodes generating high query loads. PRR nodes only know of the nearest replica, whereas Tapestry nodes enable selection from a set of replicas (for example to retrieve the most up to date). To detect routing faults, Tapestry uses TCP timeouts and UDP heartbeats for detection, sequential secondary neighbours for rerouting, and a 'second chance' window so that recovery can occur without the overhead of a full node insertion. Tapestry's dependability has been measured on a testbed of about 100 machines and on simulations of about 1000 nodes. Successful routing rates and maintenance bandwidths were measured during instantaneous failures and ongoing churn [31].

Pastry, like Tapestry, uses Plaxton-like prefix routing [2]. As in Tapestry, Pastry nodes maintain O(log N) neighbours and route to a target in O(log N) hops. Pastry differs from Tapestry only in the method by which it handles network locality and replication [2]. Each Pastry node maintains a 'leaf set' and a 'routing table'. The leaf set contains $l/2$ node IDs on either side of the local node ID in the node ID space. The routing table, in row r column c, points to the node ID with the same r-digit prefix as the local node, but with an r+1 digit of c. A Pastry node periodically probes leaf set and routing table nodes, with periodicity of T_{ls} and T_{rt} and a timeout T_{out}. Mahajan, Castry et al. analysed the reliability versus maintenance cost tradeoffs in terms of the parameters l, T_{ls}, T_{rt}, and T_{out} [288]. They concluded that earlier concerns about excessive maintenance cost in a churning P2P network were unfounded, but suggested followup work for a wider range of reliability targets, maintenance costs and probe periods. Rhea Geels et al. concluded that existing DHTs fail at high churn rates [289]. Building on a Pastry implementation from Rice University, they found that most lookups fail to complete when there is excessive churn. They conjectured that short-lived nodes often leave the network with lookups that have not yet timed out, but no evidence was provided to confirm the theory. They identified three design issues that affect DHT performance under churn: reactive versus periodic recovery of peers; lookup timeouts; and choice of nearby neighbours. Since reactive recovery was found to add traffic to already congested links, the authors used periodic recovery in their design. For lookup timeouts, they advocated an exponentially weighted moving average of each neighbour's response time, over alternative fixed timeout or 'virtual coordinate' schemes. For selection of nearby neighbours, they found that 'global sampling' was more effective than simply sampling a 'neighbour's neighbours' or 'inverse neighbours'. Castro, Costa et al. have refuted the suggestion that DHTs cannot cope with high churn rates [290]. By implementing methods for continuous detection and repair, their MSPastry implementation achieved shorter routing paths and a maintenance overhead of less than half a message per second per node.

There have been more recent proposals based on these early Plaxton-like schemes. Kademlia uses a bit-wise exclusive or (XOR) metric[4] for the 'distance' between 160 bit node identifiers [45]. Each node keeps a list of contact nodes for each section of the node space that is between 2^i and 2^{i+1} from itself ($0 \leq i < 160$). Longer-lived nodes are deliberately given preference on this list – it has been found in Gnutella that the longer a node has been active, the more likely it is to remain active. Like Kademlia, Willow uses the XOR metric [32]. It implements a Tree Maintenance Protocol to 'zipper' together broken segments of a tree. Where other schemes use DHT routing to inefficiently add new peers, Willow can merge disjoint or broken trees in O(log N) parallel operations.

3.4. Rings

Chord is the prototypical DHT ring, so we first sketch its operation. Chord maps nodes and keys to an identifier ring [7, 34]. Chord supports one main operation: find a node with the given key. It uses Consistent Hashing (Section 3.1) to minimize disruption of keys when nodes join and leave the network. However, Chord peers need only track O(log N) other peers, not all peers as in the original consistent hashing proposal [49]. It enables concurrent node insertions and deletions, improving on PRR. Compared to Pastry, it has a simpler join protocol. Each Chord peer tracks its predecessor, a list of successors and a finger table. Using the finger table, each hop is at least half the remaining distance around the ring to the target node, giving an average[5] lookup hop count of $(\frac{1}{2})\log_2 N$. Each Chord node runs a periodic stabilization routine that updates predecessor and successor pointers to cater for newly added nodes. All successors of a given node need to fail for the ring to fail. Although a node departure could be treated the same as a failure, a departing Chord node first notifies the predecessor and successors, so as to improve performance.

In their definitive paper, Chord's inventors critiqued its dependability under churn [34]. They provided proofs on the behaviour of the Chord network when nodes in a stable network fail, stressing that such proofs are inadequate in the general case of a perpetually churning network. An earlier paper had posed the question, "For lookups to be successful during churn, how regularly do the Chord stabilization routines need to run?" [291]. Stoica, Morris et al. modeled a range of node join/departure rates and stabilization periods for a Chord network of 1000 nodes. They measured the number of timeouts (caused by a finger pointing to a departed node) and lookup failures (caused by nodes that temporarily point to the wrong successor during churn). They also modelled the 'lookup stretch', the ratio

[4] To be more precise, Maymounkov and Mazieres make comparison with Pastry's first routing phase, saying that Pastry's second phase uses numeric difference.

[5] For r successors, the average hop count is more accurately expressed as $(\frac{1}{2})\log_2 N - (\frac{1}{2})\log_2(r) + 1$

of the Chord lookup time to optimal lookup time on the underlying network. They demonstrated the latency advantage of recursive lookups over iterative lookups, but there remains room for delay reduction. For further work, the authors proposed to improve resilience to network partitions, using a small set of known nodes or 'remembered' random nodes. To reduce the number of messages per lookup, they suggested an increase in the size of each step around the ring, accomplished via a larger number of fingers at each node. Much of the paper assumed independent, equally likely node failures. Analysis of correlated node failures, caused by massive site or backbone failures, will be more important in some deployments. The paper did not attempt to recommend a fixed optimal stabilization rate. Liben-Nowell, Balakrishnan et al. had suggested that optimum stabilization rate might evolve according to measurements of peers' behaviour [291] – such a mechanism has yet to be devised.

Alima, El-Ansary et al. considered the communication costs of Chord's stabilization routines, referred to as 'active correction', to be excessive [292]. Two other robustness issues also motivated their Distributed K-ary Search design, which is similar to Chord. Firstly, the total system should evolve for an optimum balance between the number of peers, the lookup hopcount and the size of the routing table. Secondly, lookups should be reliable – P2P algorithms should be able to guarantee a successful lookup for key/value pairs that have been inserted into the system. A similar lookup correctness issue was raised elsewhere by one of Chord's authors, "Is it possible to augment the data structure[6] to work even when nodes (and their associated finger lists) just disappear?" [293] Alima, El-Ansary et al. asserted that P2Ps using active correction, like Chord, Pastry and Tapesty, are unable to give such a guarantee. They propose an alternate 'correction-on-use' scheme, whereby expired routing entries are corrected by information piggybacking lookups and insertions. A prerequisite is that lookup and insertion rates are significantly higher than node arrival, departure and failure rates. Correct lookups are guaranteed in the presence of simultaneous node arrivals or up to *f* concurrent node departures, where *f* is configurable.

3.5. Tori

Ratnasamy, Francis et al. developed the Content-Addressable Network (CAN), another early DHT widely referenced alongside Tapestry, Pastry and Chord [8, 294]. It is arranged as a virtual *d*-dimensional Cartesian coordinate space on a d-torus. Each node is responsible for a zone in this coordinate space. The designers used a heuristic thought to be important for large, churning P2P networks: keep the number of neighbours independent of system size. Consequently, its design differs significantly from Pastry, Tapestry and Chord. Whereas they have O(logN) neighbours per node and O(logN) hops per lookup, CAN has O(*d*) neighbours and O($d\mathrm{n}^{1/d}$) hop-count.

[6] The question was posed in the context of a nearest neighbour search algorithm, a proposed Chord extension.

When CAN's system-wide parameter *d* is set to log(N), CAN converges to their profile. If the number of nodes grows, a major rearrangement of the CAN network may be required [151]. The CAN designers considered building on PRR, but opted for the simple, low-state-per-node CAN algorithm instead. They had reasoned that a PRR-based design would not perform well under churn, given node departures and arrivals would affect a logarithmic number of nodes [8].

There have been preliminary assessments of CAN's resilience. When a node leaves the CAN in an orderly fashion, it passes its own Virtual ID (VID), its neighbours' VIDs and IP addresses, and its key/value pairs to a takeover node. If a node leaves abruptly, its neighbours send recovery messages towards the designated takeover node. CAN ensures the recovery messages reach the takeover node, even if nodes die simultaneously, by maintaining a VID chain with Chord's stabilization algorithm. Some initial 'proof of concept' resilience simulations were run using the Network Simulator (ns) [295] for up to a few hundred nodes. Average hopcounts and lookup failure probabilities were plotted against the total number of nodes, for various node failure rates [8]. The CAN team documented several open research questions pertaining to state/hopcount tradeoffs, resilience, load, locality and heterogeneous peers [44, 294].

3.6. Butterflies

Viceroy approximates a butterfly network [46]. It generally has constant degree[7] like CAN. Like Chord, Tapesty and Pastry, it has logarithmic diameter. It improves on these systems, inasmuch as its diameter is better than CAN and its degree is better than Chord, Tapestry and Pastry. As with most DHTs, it utilizes Consistent Hashing. When a peer joins the Viceroy network, it takes a random but permanent 'identity' and selects its 'level' within the network. Each peer maintains general ring pointers ('predecessor' and 'successor'), level ring pointers ('nextonlevel' and 'prevonlevel') and butterfly pointers ('left', 'right' and 'up'). When a peer departs, it normally passes its key pairs to a successor, and notifies other peers to find a replacement peer.

The Viceroy paper scoped out the issue of robustness. It explicitly assumed that peers do not fail [46]. It assumed that join and leave operations do not overlap, so as to avoid the complication of concurrency mechanisms like locking. Kaashoek and Karger were somewhat critical of Viceroy's complexity [37]. They also pointed to its fault tolerance blindspot. Li and Plaxton suggested that such constant-degree algorithms deserve further consideration [47]. They offered several pros and cons. The limited degree may increase the risk of a network partition, or inhibit

[7] Viceroy's expected degree is a constant. However, its high probability bound is O(log n). For a very small number of nodes, degree is Ω(log n).

use of local neighbours (for the simple reason that there are less of them). On the other hand, it may be easier to reason about the correctness of fixed-degree networks. One of the Viceroy authors has since proposed constant-degree peers in a two-tier, locality-aware DHT [296] – the lower degree maintained by each lower-tier peer purportedly improves network adaptability. Another Viceroy author has since explored an alternative bounded-degree graph for P2P, namely the de Bruijn graph [297].

3.7. de Bruijn Graphs

De Bruijn graphs have had numerous refinements since their inception [298, 299]. Schlumberger was the first to use them for networking [300]. Two research teams independently devised the 'generalized' de Bruijn graph that accommodates a flexible number of nodes in the system [301, 302]. Rowley and Bose studied fault-tolerant rings overlaid on the de Bruijn graph [303]. Lee, Liu et al. devised a two-level de Bruijn hierarchy, whereby clusters of local nodes are interconnected by a second-tier ring [304].

Many of the algorithms discussed previously are 'greedy' in that each time a query is forwarded, it moves closer to the destination. Unfortunately, greedy algorithms are generally suboptimal – for a given degree, the routing distance is longer than necessary [305]. Unlike these earlier P2P designs, de Bruijn graphs of degree *k* achieve an asymptotically optimal diameter $\log_k n$, where n is the number of nodes in the system and *k* can be varied to improve resilience. If there are O(log(n)) neighbours per node, the de Bruijn hop count is O(log n/log log n). To illustrate de Bruijn's practical advantage, consider a network with one million nodes of degree 20: Chord has a diameter of 20, while de Bruijn has a diameter of 5 [36]. In 2003, there were a quick succession of de Bruijn proposals – D2B [306], Koorde [37], Distance Halving [132, 297] and the Optimal Diameter Routing Infrastructre (ODRI) [36].

Fraigniaud and Gauron began the D2B design by laying out an informal problem statement: keys should be evenly distributed; lookup latency should be small; traffic load should be evenly distributed; updates of routing tables and redistribution of keys should be fast when nodes join or leave the network. They defined a node's "congestion" to be the probability that a lookup will traverse it. Apart from its optimal de Bruijn diameter, they highlighted D2B's merits: a constant expected update time when nodes (O(log n) w.h.p[8].); the expected node congestion is O((logn)/n) (O(($\log^2$n)/n) w.h.p.) [306]. D2B's resilience was discussed only in passing.

[8] W.h.p. With high probability $1 - n^{-\varepsilon}$

Koorde extends Chord to attain the optimal de Bruijn degree/diameter tradeoff above [37]. Unlike D2B, Koorde does not constrain the selection of node identifiers. Also unlike D2B, it caters for concurrent joins, by extension of Chord's functionality. Kaashoek and Karger investigated Koorde's resilience to a rather harsh failure scenario: "in order for a network to stay connected when all nodes fail with probability of ½, some nodes must have degree Ω(log n)" [37]. They sketched a mechanism to increase Koorde's degree for this more stringent fault tolerance, losing de Bruijn's constant degree advantage. Similarly, to achieve a constant-factor load balance, Koorde would have to sacrifice its degree optimality. They suggested that the ability to trade the degree, and hence the maintenance overhead, against the expected hop count may be important for churning systems. They also identified an open problem: find a load-balanced, degree optimal DHT. Datta, Girdzijauskas et al. showed that for arbitrary key distributions, de Bruijn graphs fail to meet the dual goals of load balancing and search efficiency [307]. They posed the question, "(Is there) a constant routing table sized DHT which meets the conflicting goals of storage load balancing and search efficiency for an arbitrary and changing key distribution?"

Distance Halving was also inspired by de Bruijn [297] and shares its optimal diameter. Naor and Wieder argued for a two-step "continuous-discrete" approach for its design. The correctness of its algorithms is proven in a continuous setting. The algorithms are then mapped to a discrete space. The source x and target y are points on the continuous interval [0,1). Data items are hashed to this same interval. σ is a string which determines how messages leave any point on the ring: if bit *t* of the string is 0, the left leg is taken; if it is 1, the right leg is taken. σ increases by one bit each hop, giving a sequence by which to step around the ring. A lookup has two phases. In the first, the lookup message containing the source, target and the random string hops toward the midpoint of the source and target. On each hop, the distance between $\sigma_t(x)$ and $\sigma_t(y)$ is halved, by virtue of the specific 'left' and 'right' functions. In the second phase, the message steps 'backward' from the midpoint to the target, removing the last bit in σ_t at each hop. 'Join' and 'leave' algorithms were outlined but there was no consideration of recovery times or message load on churn. Using the Distance Halving properties, the authors devised a caching scheme to relieve congestion in a large P2P network. They have also modified the algorithm to be more robust in the presence of random faults [132].

Solid comparisons of DHT resilience are scarce, but Loguinov, Kumar et al. give just that in their ODRI paper [36]. They compare Chord, CAN and de Bruijn in terms of routing performance, graph expansion and clustering. At the outset, they give the optimal *diameter* (the *maximum* hopcount between any two nodes in the graph) and *average*

hopcount for graphs of fixed degree. De Bruijn graphs converge to both optima, and outperform Chord and CAN on both counts. These optima impact both delay and aggregate lookup load. They present two clustering measures (edge expansion and node expansion) which are interesting for resilience. Unfortunately, after decades of de Bruijn research, they have no exact solution. De Bruijn was shown to be superior in terms of path overlap – "de Bruijn automatically selects backup paths that do not overlap with the best shortest path or with each other" [36].

3.8. Skip Graphs

Skip Graphs have been pursued by two research camps [38, 41]. They augment the earlier Skip Lists [308, 309]. Unlike earlier balanced trees, the Skip List is probabilistic – its insert and delete operations do not require tree rearrangements and so are faster by a constant factor. The Skip List consists of layers of ordered linked lists. All nodes participate in the bottom layer 0 list. Some of these nodes participate in the layer 1 list with some fixed probability. A subset of layer 1 nodes participate in the layer 2 list, and so on. A lookup can proceed quickly through the list by traversing the sparse upper layers until it is close to, or at, the target. Unfortunately, nodes in the upper layers of a Skip List are potential hot spots and single points of failure. Unlike Skip Lists, Skip Graphs provide multiple lists at each level for redundancy, and every node participates in one of the lists at each level.

Each node in a Skip Graph has $\Theta(\log n)$ neighbours on average, like some of the preceding DHTs. The Skip Graph's primary edge over the DHTs is its support for prefix and proximity search. DHTs hash objects to a random point in the graph. Consequently, they give no guarantees over where the data is stored. Nor do they guarantee that the path to the data will stay within the one administration as far as possible [38]. Skip graphs, on the other hand, provide for location-sensitive name searches. For example, to find the document *docname* on the node *user.company.com*, the Skip Graph might step through its ordered lists for the prefix *com.company.user* [38]. Alternatively, to find an object with a numeric identifier, an algorithm might search the lowest layer of the Skip Graph for the first digit, the next layer for the next digit, in the same vein until all digits are resolved. Being ordered, Skip Graphs also facilitate range searches. In each of these examples, the Skip Graph can be arranged such that the path to the target, as far as possible, stays within an administrative boundary. If one administration is detached from the rest of the Skip Graph, routing can continue within each of the partitions. Mechanisms have been devised to merge disconnected segments [157], though at this stage, segments are remerged one at a time. A parallel merge algorithm has been flagged for future work.

The advantages of Skip Graphs come at a cost. To be able to provide range queries and data placement flexibility, Skip Graph nodes require many more pointers than their DHT counterparts. An increased number of pointers implies increased maintenance traffic. Another shortcoming of at least one of the early proposals was that no algorithm was given to assign keys to machines. Consequently, there are no guarantees on system-wide load balancing or on the distance between adjacent keys [100]. Aspnes, Kirsch et al. have recently devised a scheme to reduce the inter-machine pointer count from $O(m \log m)$, where m is the number of data elements, to $O(n \log n)$, where n is the number of nodes [100]. They proposed a two-layer scheme – one layer for the Skip Graph itself and the second 'bucket layer'. Each machine is responsible for a number of buckets and each bucket elects a representative key. Nodes locally adjust their load. They accept additional keys if they are below their threshold or disperse keys to nearby nodes if they are above threshold. There appear to be numerous open issues: simulations have been done but analysis is outstanding; mechanisms are required to handle the arrival and departure of nodes; there were only brief hints as to how to handle nodes with different capacities.

4. Semantic Index

Semantic indexes capture object relationships. While the semantic-free methods (DHTs) have firmer theoretic foundations and guarantee that a key can be found if it exists, they do not on their own capture the relationships between the document name and its content or metadata. Semantic P2P designs do. However, since their design is often driven by heuristics, they may not guarantee that scarce items will be found.

So what might the semantically indexed P2Ps add to an already crowded field of distributed information architectures? At one extreme there are the distributed relational database management systems (RDBMSs), with their strong consistency guarantees [264]. They provide strong data independence, the flexibility of SQL queries and strong transactional semantics – Atomicity, Consistency, Isolation and Durability (ACID) [310]. They guarantee that the query response is complete – all matching results are returned. The price is performance. They scale to perhaps 1000 nodes, as evidenced in Mariposa [311, 312], or require query caching front ends to constrain the load [264]. Database research has "arguably been cornered into traditional, high-end, transactional applications" [72]. Then there are distributed file systems, like the Network File System (NFS) or the Serverless Network File Systems (xFS), with little data independence, low-level file retrieval interfaces and varied consistency [264]. Today's eclectic mix of Content Distribution Networks (CDNs) generally deload primary servers by redirecting web requests to a nearby replica. Some intercept the HTTP requests at the DNS level and then use consistent hashing to find a replica

[23]. Since this same consistent hashing was a forerunner to the DHT approaches above, CDNs are generally constrained to the same simple key lookups.

The opportunity for semantically indexed P2Ps, then, is to provide:

a) graduated data independence, consistency and query flexibility, and

b) probabilistically complete query responses, across

c) very large numbers of low-cost, geographically distributed, dynamic nodes.

4.1. Keyword Lookup

P2P keyword lookup is best understood by considering the structure of the underlying index and the algorithms by which queries are routed over that index. Figure 3 summarizes the following paragraphs by classifying the keyword query algorithms, index structures and metrics. The research has largely focused on *scalability*, not *dependability*. There have been very few studies that quantify the impact of network churn. One exception is the work by Chawathe et al. on the Gia system [61]. Gia's combination of algorithms from Figure 3 (receiver-based flow control, biased random walk and one-hop replication) gave 2-4 orders of magnitude improvement in query success rates in churning networks.

Perhaps the most widely referenced P2P system for simple keyword match is Gnutella [4]. Gnutella queries contain a string of keywords. Gnutella peers answer when they have files[9] whose names contain all the keywords. As discussed in Section 2.1, early versions of Gnutella did not forward the document index. Queries were flooded and peers searched their own local indexes for filename matches. An early review highlighted numerous areas for improvement [65]. It was estimated that the query traffic alone from 50,000 early-generation Gnutella nodes would amount to 1.7% of the total U.S. internet backbone traffic at December 2000 levels. It was speculated that high degree Gnutella nodes would impede dependability. An unnecessarily high percentage of Gnutella traffic crossed Autonomous System (AS) boundaries – a locality mechanism may have found suitable nearby peers.

Fortunately, there have since been numerous enhancements within the Gnutella Developer Forum. At the time of writing, it has been reported that Gnutella has almost 350,000 unique hosts, of which nearly 90,000 accept incoming connections [317]. One of the main improvements is that an index of filename keywords, called the Query Routing

[9] The Gnutella 0.6 specification only provides semantics for finding plain files, but hints that Gnutella could store other resources, like cryptographic keys or meta-information.

Table (QRT), can now be forwarded from 'leaf peers' to its 'ultrapeers' [240]. Ultrapeers can then ensure that the leaves only receive queries for which they have a match, dramatically reducing the query traffic at the leaves. Ultrapeers can have connections to many leaf nodes (~10-100) and a small number of other ultrapeers (<10) [240]. Originally, a leaf node's QRT was not forwarded by the parent ultrapeer to other ultrapeers. More recently, there has been a proposal to distribute aggregated QRTs amongst ultrapeers [318]. To further limit traffic, QRTs are compressed by hashing, according to the Query Routing Protocol (QRP) specification [261]. This same specification claims QRP may reduce Gnutella traffic by orders of magnitude, but cautions that simulation is required before mass deployment. A known shortcoming of QRP was that the extent of query propagation was independent of the popularity of the search terms. The Dynamic Query Protocol addressed this [319]. It required leaf nodes to send single queries to high-degree ultrapeers which adjust the queries' time-to-live (TTL) bounds according to the number of received query results. An earlier proposal, called the Gnutella UDP Extension for Scalable Searches (GUESS) [314], similarly aimed to reduce the number of queries for widely distributed files. GUESS reuses the *non-forwarding* idea (Section 2). A GUESS peer repeatedly queries single ultrapeers with a TTL of 1, with a small timeout on each query to limit load. It chooses the number of iterations and selects ultrapeers so as to satisfy its search needs. For adaptability, a small number of experimental Gnutella nodes have implemented eXtensible Markup Language (XML) schemas for richer queries [320, 321]. None of the above Gnutella proposals explicitly assess robustness.

The broader research community has recently been leveraging aspects of the Gnutella design. Lv, Ratnasamy et al. exposed one assumption implicit in some of the early DHT work – that designs "such as Gnutella are inherently not scalable, and therefore should be abandoned" [66]. They argued that by making better use of the more powerful peers, Gnutella's scalability issues could be alleviated. Instead of its flooding mechanism, they used random walks. Their preliminary design to bias random walks towards high capacity nodes did not go as far as the ultrapeer proposals in that the indexes did not move to the high capacity nodes. Chawathe, Ratnasamy et al. chose to extend the Gnutella design with their Gia system, in response to the perceived shortcomings of DHTs in Section 1.2 [61]. Compared to the early Gnutella designs, they incorporated several novel features. They devise a topology adaptation algorithm so that most peers are attached to high-degree peers. They use a random walk search algorithm, in lieu of flooding, and bias the query load towards higher-degree peers. For 'one-hop replication', they require all nodes keep pointers to content on adjacent peers. To implement a receiver-controlled token-based flow control, a peer must have a token from its neighbouring peer before it sends a query to it. Chawathe, Ratnasamy et al. show by

simulations that the combination of these features provides a scalability improvement of three to five orders of magnitude over Gnutella "while retaining significant robustness". The main robustness metrics they used were the 'collapse point' query rate (the per node query rate at which the successful query rate falls below 90%) and the average hop-count immediately prior to collapse. Their comparison with Gnutella did not take into account the Gnutella enhancements above – this was left as future work. Castro, Costa and Rowstron argued that if Gnutella were built on top of a structured overlay, then both the query and overlay maintenance traffic could be reduced [239]. Yang, Vinograd et al. explore various policies for peer selection in the GUESS protocol, since the issue is left open in the original proposal [245]. For example, the peer initiating the query could choose peers that have been "most recently used" or that have the "most files shared". Various policy pitfalls are identified. For example, good peers could be overloaded, victims of their own success. Alternatively, malicious peers could encourage the querying peer to try inactive peers. They conclude that a "most results" policy gives the best balance of robustness and efficiency. Like Castro, Costa and Rowstron, they concentrated on the static network scenario. Cholvi, Felber et al. very briefly describe how similar "least recently used" and "most often used" heuristics can be used by a peer to select peer 'acquaintances' [313]. They were motivated by the congestion associated with Gnutella's TTL-limited flooding. Recognizing that the busiest peers can quickly become overloaded central hubs for the entire network, they limit the number of acquaintances for any given peer to 25. They sketch a mechanism to decrement a query's TTL multiple times when it traverses "interested peers". In summary, these Gnutella-related investigations are characterized by a bias for high degree peers and very short directed query paths, a disdain for flooding, and concern about excessive load on the 'better' peers. Generally, the robustness analysis for dynamic networks (content updates and node arrivals/departures) remains open.

One aspect of P2P keyword search systems has received particular attention: should the index be partitioned by document or by keyword? The issue affects scalability. To be partitioned by document, each node has a local index of documents for which it is responsible. Gnutella is a prime example. Queries are generally flooded in systems partitioned by document. On the other hand, a peer may assume responsibility for a set of keywords. The peer uses an inverted list to find a matching document, either locally or at another peer. If the query contains several keywords, inverted lists may need to be retrieved from several different peers to find the intersection [21]. The initial assessment by Li, Loo et al. was that the partition-by-document approach was superior [210]. For one scenario of a full-text web search, they estimated the communications costs to be about six times higher than the feasible budget. However, wanting to exploit prior work on inverted list intersection, they studied the partition-by-

keyword strategy. They proposed several optimizations which put the communication costs for a partition-by-keyword system within an order of magnitude of feasibility. There had been a couple of prior papers that suggested partitioned-by-keyword designs incorporate DHTs to map keywords to peers [316, 322]. In Gnawali's Keyword-set Search System (KSS), the index is partitioned by *sets* of keywords [316]. Terpstra, Behnel et al. point out that by keeping keyword pairs or triples, the number of lists per document in KSS is squared or tripled [323]. Shi, Guangwen et al. interpreted the approximations of Li, Loo et al. to mean that neither approach is feasible on its own [21]. Their Multi-Level Partitioning (MLP) scheme incorporates both partitioning approaches. They arrange nodes into a group hierarchy, with all nodes in the single 'level 0' group, and with the same nodes sub-divided into k logical subgroups on 'level 1'. The subgroups are again divided, level by level, until level l. The inverted index is partitioned by document between groups and by keyword within groups. MLP avoids the query flooding normally associated with systems partitioned by document, since a small number of nodes in each group process the query. It reduces the bandwidth overheads associated with inverted list intersection in systems partitioned solely by keyword, since groups can calculate the intersection independently over the documents for which they are responsible. MLP was overlaid on SkipNet, per Section 3.8 [38]. Some initial analyses of communications costs and query latencies were provided.

Much of the research above addresses *partial* keyword search. Daswani et al. highlighted the open problem of efficient, *comprehensive* keyword search [25]. How can exhaustive searches be achieved without flooding queries to every peer in the network? Terpstra, Behnel et al. couched the keyword search problem in rendezvous terms: dynamic keyword queries need to 'meet' with static document lists [323]. Their Bitzipper scheme is partitioned by document. They improved on full flooding by putting document metadata on $2\sqrt{n}$ nodes and forwarding queries through only $6\sqrt{n}$ nodes. They reported that Bitzipper nodes need only $1/166^{th}$ of the bandwidth of full-flooding Gnutella nodes for an exhaustive search. An initial comparison of query load was given. There was little consideration of either static or dynamic resilience, that is, of nodes failing, of documents continually changing, or of nodes continually joining and leaving the network.

4.2. Peer Information Retrieval

The field of Information Retrieval (IR) has matured considerably since its inception in the 1950s [324]. A taxonomy for IR models has been formalized [242]. It consists of four elements: a representation of documents in a collection; a representation of user queries; a framework describing relationships between document representations

and queries; and a ranking function that quantifies an ordering amongst documents for a particular query. Three main issues motivate current IR research – information relevance, query response time, and user interaction with IR systems. The dominant IR trends for searching large text collections are also threefold [242]. The size of collections is increasing dramatically. More complicated search mechanisms are being found to exploit document structure, to accommodate heterogeneous document collections, and to deal with document errors. Compression is in favour – it may be quicker to search compact text or retrieve it from external devices. In a *distributed* IR system, query processing has four parts. Firstly, particular collections are targeted for the search. Secondly, queries are sent to the targeted collections. Queries are then evaluated at the individual collections. Finally results from the collections are collated.

So how do P2P networks differ from distributed IR systems? Bawa, Manku et al. presented four differences [62]. They suggested that a P2P network is typically larger, with tens or hundreds of thousands of nodes. It is usually more dynamic, with node lifetimes measured in hours. They suggested that a P2P network is usually homogeneous, with a common resource description language. It lacks the centralized "mediators" found in many IR systems, that assume responsibility for selecting collections, for rewriting queries, and for merging ranked results. These distinctions are generally aligned with the peer characteristics in Section 1. One might add that P2P nodes display more symmetry – peers are often both information consumers and producers. Daswani, Garcia-Molina et al. pointed out that, while there are IR techniques for ranked keyword search at moderate scale, research is required so that ranking mechanisms are efficient at the larger scale targeted by P2P designs [25]. Joseph and Hoshiai surveyed several P2P systems using metadata techniques from the IR toolkit [60]. They described an assortment of IR techniques and P2P systems, including various metadata formats, retrieval models, bloom filters, DHTs and trust issues.

In the ensuing paragraphs, we survey P2P work that has incorporated information retrieval models, particularly the Vector Model and the Latent Semantic Indexing Model. We omit the P2P work based on Bayesian models. Some have pointed to such work [60], but it made no explicit mention of the model [325]. One early paper on P2P content-based image retrieval also leveraged the Bayesian model [326]. For the former two models, we briefly describe the design, then try to highlight robustness aspects. On robustness, we are again stymied for lack of prior work. Indeed, a search across all proceedings of the Annual ACM Conference on Research and Development in Information Retrieval for the words "reliable", "available", "dependable" or "adaptable" did not return any results at

the time of writing. In contrast, a standard text on distributed database management systems [327] contains a whole chapter on reliability. IR research concentrates on performance measures. Common performance measures include recall, the fraction of the relevant documents which has been retrieved, and precision, the fraction of the retrieved documents which is relevant [242]. Ideally, an IR system would have high recall and high precision. Unfortunately techniques favouring one often disadvantage the other [324].

4.2.1. Vector Model

The vector model [328] represents both documents and queries as term vectors, where a term could be a word or a phrase. If a document or query has a term, the weight of the corresponding dimension of the vector is non-zero. The similarity of the document and query vectors gives an indication of how well a document matches a particular query.

The weighting calculation is critical across the retrieval models. Amongst the numerous proposals for the probabilistic and vector models, there are some commonly recurring weighting factors [324]. One is term frequency. The more a term is repeated in a document, the more important the term is. Another is inverse document frequency. Terms common to many documents give less information about the content of a document. Then there is document length. Larger documents can bias term frequencies, so weightings are sometimes normalized against document length. The expression “TFIDF weighting” refers to the collection of weighting calculations that incorporate term frequency and inverse document frequency, not just to one. Two weighting calculations have been particularly dominant – Okapi [329] and pivoted normalization [330]. A distributed version of Google’s Pagerank algorithm has also been devised for a P2P environment [331]. It allows incremental, ongoing Pagerank calculations while documents are inserted and deleted.

A couple of early P2P systems leveraged the vector model. Building on the vector model, PlanetP divided the ranking problem into two steps [215]. In the first, peers are ranked for the probability that they have matching documents. In the second, higher priority peers are contacted and the matching documents are ranked. An Inverse Peer Frequency, analogous to the Inverse Document Frequency, is used to rank relevant peers. To further constrain the query traffic, PlanetP contacts only the first group of m peers to retrieve a relevant set of documents. In this way, it repeatedly contacts groups of m peers until the top k document rankings are stable. While the PlanetP designers first quantified recall and precision, they also considered reliability. Each PlanetP peer has a global index with a list of all other peers, their IP addresses, and their Bloom filters. This large volume of shared information needs to be

maintained. Klampanos and Jose saw this as PlanetP's primary shortcoming [332]. Each Bloom filter summarized the set of terms in the local index of each peer. The time to propagate changes, be they new documents or peer arrivals/departures, was studied by simulation for up to 1000 peers. The reported propagation times were in the hundreds of seconds. Design workarounds were required for PlanetP to be viable across slower dial-up modem connections. For future work, the authors were considering some sort of hierarchy to scale to larger numbers of peers.

A second early system using the vector model is the Fault-tolerant, Adaptive, Scalable Distributed search engine [263], which extended the Freenet design (Section 2.3) for richer queries. The original Freenet design could find a document based on a globally unique identifier. Kronfol's design added the ability to search, for example, for documents about "apples AND oranges NOT bananas". It uses a TFIDF weighting scheme to build a document's term vector. Each peer calculates the similarity of the query vector and local documents and forwards the query to the best downstream peer. Once the best downstream peer returns a result, the second-best peer is tried, and so on. Simulations with 1000 nodes gave an indication of the query path lengths in various situations - when routing queries in a network with constant rates of node and document insertion, when bootstrapping the network in a "worst-case" ring topology, or when failing randomly and specifically selected peers. Kronfol claimed excellent average-case performance – less than 20 hops to retrieve the same top *n* results as a centralized search engine. There were, however, numerous cases where the worst-case path length was several hundred hops in a network of only 1000 nodes.

In parallel, there have been some P2P designs based on the vector model from the University of Rochester – pSearch[10] [9, 333] and eSearch [334]. The early pSearch paper suggested a couple of retrieval models, one of which was the Vector Space Model, to search only the nodes likely to have matching documents. To obtain approximate global statistics for the TFIDF calculation, a spanning tree was constructed across a subset of the peers. For the *m* top terms, the term-to-document index was inserted into a Content-Addressable Network [294]. A variant which mapped terms to document *clusters* was also suggested. eSearch is a hybrid of the partition-by-document and partition-by-term approaches seen in the previous section. eSearch nodes are primarily partitioned by term. Each is responsible for the inverted lists for some top terms. For each document in the inverted list, the node stores the complete term list. To reduce the size of the index, the complete term lists for a document are only kept on nodes

[10] The pSearch design had earlier been proposed under the name PeerSearch

that are responsible for top terms in the document. eSearch uses the Okapi term weighting to select top terms. It relies on the Chord DHT [34] to associate terms with nodes storing the inverted lists. It also uses automatic query expansion. This takes the significant terms from the top document matches and automatically adds them to the user's query to find additional relevant documents. The eSearch performance was quantified in terms of search precision, the number of retrieved documents, and various load-balancing metrics. Compared to the more common proposals for partitioning by keywords, eSearch consumed 6.8 times the storage space to achieve faster search times.

4.2.2. Latent Semantic Indexing

Another retrieval model used in P2P proposals is Latent Semantic Indexing (LSI) [335]. Its key idea is to map both the document and query vectors to a concept space with lower dimensions. The starting point is a $t*N$ weighting matrix, where t is the total number of indexed terms, N is the total number of documents, and the matrix elements could be TFIDF rankings. Using singular value decomposition, this matrix is reduced to a smaller number of dimensions, while retaining the more significant term-to-document mappings. Baeza-Yates and Ribeiro-Neto suggested that LSI's value is a novel theoretic framework, but that its practical performance advantage for real document collections had yet to be proven [242]. pSearch incorporated LSI [9]. By placing the indices for semantically similar documents close in the network, Tang, Xu et al. touted significant bandwidth savings relative to the early full-flooding variant of Gnutella [333]. They plotted the number of nodes visited by a query. The also explored the tradeoff with accuracy, the percentage match between the documents returned by the distributed pSearch algorithm and those from a centralized LSI baseline. In a more recent update to the pSearch work, Tang, Dwarkadas et al. summarized LSI's shortcomings [336]. Firstly, for large document collections, its retrieval quality is inherently inferior to Okapi. Secondly, singular value decomposition consumes excessive memory and computation time. Consequently, the authors used Okapi for searching while retaining LSI for indexing. With Okapi, they selected the next node to be searched and selected documents on searched nodes. With LSI, they ensured that similar documents are clustered near each other, thereby optimizing the network search costs. When retrieving a small number of top documents, the precision of LSI+Okapi approached that of Okapi. However, if retrieving a large number of documents, the LSI+Okapi precision is inferior. The authors want to improve this in future work.

5. Queries

Database research suggests *directions* for P2P research. Hellerstein observed that, while work on fast P2P indexes is well underway, P2P query optimization remains a promising topic for future research [23]. Kossman reviewed the state of the art of distributed query processing, highlighting areas for future research: simulation and query optimization for networks of tens of thousands of servers and millions of clients; non-relational data types like XML, text and images; and partial query responses since on the Internet "failure is the rule rather than the exception" [19]. A primary motivation for the P2P system, PIER, was to scale from the largest database systems of a few hundred nodes to an Internet environment in which there are over 160 million nodes [22]. Litwin and Sahri have also considered ways to combine distributed hashing, more specifically the Scalable Distributed Data Structures, with SQL databases, claiming to be first to implement scalable distributed database partitioning [337]. Motivated by the lack of transparent distribution in current distributed databases, they measure query execution times for Microsoft SQL servers aggregated by means of an SDDS layer. One of their starting assumptions was that it is too challenging to change the SQL query optimizer.

Database research also suggests the *approach* to P2P research. Researchers of database query optimization were divided between those looking for optimal solutions in special cases and those using heuristics to answer all queries [338]. Gribble et al. cast query optimization in terms of the data placement problem, which is to "distribute data and work so the full query workload is answered with lowest cost under the existing bandwidth and resource constraints" [231]. They pointed out that even the static version of this problem is NP-complete in P2P networks. Consequently, research on massive, dynamic P2P networks will likely progress using both strategies of early database research - heuristics and special-case optimizations.

If P2P networks are going to be adaptable, if they are to support a wide range of applications, then they need to accommodate many query types [72]. Up to this point, we have reviewed queries for keys (Section 3) and keywords (Sections 4.1and 4.2). Unfortunately, a major shortcoming of the DHTs in Sections 3.3 to 3.7 is that they primarily support exact-match, single-key queries. Skip Graphs support range and prefix queries, but not aggregation queries. Here we probe below the language syntax to identify the open research issues associated with more expressive P2P queries [25]. Triantafillou and Pitoura observed the disparate P2P designs for different types of queries and so outlined a unifying framework [76]. To classify queries, they considered the number of relations (single or multiple), the number of attributes (single or multiple) and the type of query operator. They described numerous

operators: equality, range, join and “special functions”. The latter referred to aggregation (like sum, count, average, minimum and maximum), grouping and ordering. The following sections approximately fit their taxonomy - range queries, multi-attribute queries, join queries and aggregation queries. There has been some initial P2P work on other query types - continuous queries [20, 22, 73], recursive queries [22, 74] and adaptive queries [23, 75]. For these, we defer to the primary references.

5.1. Range Queries

The support of efficient range predicates in P2P networks was identified as an important open research issue by Huebsch et al. [22]. Range partitioning has been important in parallel databases to improve performance, so that a transaction commonly needs data from only one disk or node [22]. One type of range search, longest prefix match, is important because of its prevalence in routing schemes for voice and data networks alike. In other applications, users may pose broad, inexact queries, even though they require only a small number of responses. Consequently techniques to locate similar ranges are also important [77]. Various proposals for range searches over P2P networks are summarized in Figure 4. Since the Scalable Distributed Data Structure (SDDS) has been an important influence on contemporary Distributed Hash Tables (DHTs) [49-51], we also include ongoing work on SDDS range searches.

The papers on P2P range search can be divided into those that rely on an underlying DHT (the first five entries in (Figure 4) and those that do not (the subsequent three entries). Bharambe, Agrawal et al. argued that DHTs are inherently ill-suited to range queries [84]. The very feature that makes for their good load balancing properties, randomized hash functions, works against range queries. One possible solution would be to hash ranges, but this can require a priori partitioning. If the partitions are too large, partitions risk overload. If they are too small, there may be too many hops.

Despite these potential shortcomings, there have been several range query proposals based on DHTs. If hashing ranges to nodes, it is entirely possible that overlapping ranges map to different nodes. Gupta, Agrawal et al. rely on locality sensitive hashing to ensure that, with high probability, similar ranges are mapped to the same node [77]. They propose one particular family of locality sensitive hash functions, called min-wise independent permutations. The number of partitions per node and the path length were plotted against the total numbers of peers in the system. For a network with 1000 nodes, the hop-count distribution was very similar to that of the exact-matching Chord scheme. Was it load-balanced? For the same network with 50,000 partitions, there were over two orders of magnitude variation in the number of partitions at each node (first and ninety-ninth percentiles). The Prefix Hash

Tree is a trie in which prefixes are hashed onto any DHT. The preliminary analysis suggests efficient doubly logarithmic lookup, balanced load and fault resilience [78, 79]. Andrzejak and Xu were perhaps the first to propose a mapping from ranges to DHTs [80]. They use one particular Space Filling Curve, the Hilbert curve, over a Content Addressable Network (CAN) construction (Section 3.5). They maintain two properties: nearby ranges map to nearby CAN zones; if a range is split into two sub-ranges, then the zones of the sub-ranges partition the zone of the primary range. They plot path length and load proxy measures (the total number of messages and nodes visited) for three algorithms to propagate range queries: brute force; controlled flooding and directed controlled flooding. Schmidt and Parashar also advocated Space Filling Curves to achieve range queries over a DHT [81]. However they point out that, while Andrzejak and Xu use an inverse Space Filling Curve to map a one-dimensional space to d-dimensional zones, they map a d-dimensional space back to a one-dimensional index. Such a construction gives the ability to search across multiple attributes (Section 5.2). Tanin, Harwood et al. suggested quadtrees over Chord [82], and gave preliminary simulation results for query response times.

Because DHTs are naturally constrained to exact-match, single-key queries, researchers have considered other P2P indexes for range searches. Several were based on Skip Graphs [38, 41] which, unlike the DHTs, do not necessitate randomizing hash functions and are therefore capable of range searches. Unfortunately, they are not load balanced [83]. For example, in SkipNet [48], hashing was added to balance the load - the Skip Graph could support range searches or load balancing, but not both. One solution for load-balancing relies on an increased number of 'virtual' servers [168] but, in their search for a system that can both search for ranges and balance loads, Bharambe, Agrawal et al. rejected the idea [84]. The virtual servers work assumed load imbalance stems from hashing, that is, by skewed data insertions and deletions. In some situations, the imbalance is triggered by a skewed query load. In such circumstances, additional virtual servers can increase the number of routing hops and increase the number of pointers that a Skip Graph needs to maintain. Ganesan, Bawa et al. devised an alternate method to balance load [83]. They proposed two Skip Graphs, one to index the data itself and the other to track load at each node in the system. Each node is able to determine the load on its neighbours and the most (least) loaded nodes in the system. They devise two algorithms: NBRADJUST balances load on neighbouring nodes; using REORDER, empty nodes can take over some of the tuples on heavily loaded nodes. Their simulations focus on skewed storage load, rather than on skewed query loads, but they surmise that the same approach could be used for the latter.

Other proposals for range queries avoid both the DHT and the Skip Graph. Bharambe, Agrawal et al. distinguish their Mercury design by its support for multi-attribute range queries and its explicit load balancing [84]. In Mercury, nodes are grouped into routing hubs, each of which is responsible for various query attributes. While it does not use hashing, Mercury is loosely similar to the DHT approaches: nodes within hubs are arranged into rings, like Chord [34]; for efficient routing within hubs, k long-distance links are used, like Symphony [339]. Range lookups require $O(\log^2 n/k)$ hops. Random sampling is used to estimate the average load on nodes and to find the parts of the overlay that are lightly loaded. Whereas Symphony assumed that nodes are responsible for ranges of approximately equal size, Mercury's random sampling can determine the location of the start of the range, even for non-uniform ranges [84]. P-Grid [42] does provide for range queries, by virtue of the key ordering in its tree structures. Ganesan, Bawa et al. critiqued its capabilities [83]: P-Grid assumes fixed-capacity nodes; there was no formal characterization of imbalance ratios or balancing costs; every P-Grid periodically contacts other nodes for load information.

The work on Scalable Distributed Data Structures (SDDSs) has progressed in parallel with P2P work and has addressed range queries. Like the DHTs above, the early SDDS Linear Hashing (LH*) schemes were not order-preserving [52]. To facilitate range queries, Litwin, Niemat et al. devised a Range Parititioning variant, RP* [87]. There are options to dispense with the index, to add indexes to clients and to add them to servers. In the variant without an index, every query is issued via multicasting. The other variants also use some multicasting. The initial RP* paper suggested scalability to thousands of sites, but a more recent RP* simulation was capped at 140 servers [88]. In that work, Tsangou, Ndiaye et al. investigated TCP and UDP mechanisms by which servers could return range query results to clients. The primary metrics were search and response times. Amongst the commercial parallel database management systems, they reported that the largest seems only to scale to 32 servers (SQL Server 2000). For future work, they planned to explore aggregation of query results, rather than establishing a connection between the client and every single server with a response.

All in all, it seems there are numerous open research questions on P2P range queries. How realistic is the maintenance of global load statistics considering the scale and dynamism of P2P networks? Simulations at larger scales are required. Proposals should take into account both the storage load (insert and delete messages) and the query load (lookup messages). Simplifying assumptions need to be attacked. For example, how well do the above solutions work in networks with heterogeneous nodes, where the maximum message loads and index sizes are node-dependent?

5.2. Multi-Attribute Queries

There has been some work on multi-attribute P2P queries. As late as September 2003, it was suggested that there has not been an efficient solution [76].

Again, an early significant work on multi-attribute queries over aggregated commodity nodes germinated amongst SDDSs. k-RP* [89] uses the multi-dimensional binary search tree (or k-d tree where k indicates the number of dimensions of the search index) [340]. It builds on the RP* work from the previous section and inherits their capabilities for range search and partial match. Like the other SDDSs, k-RP* indexes can fit into RAM for very fast lookup. For future work, Litwin and Neimat suggested a) a formal analysis of the range search termination algorithm and the k-d paging algorithm, b) a comparison with other multi-attribute data structures (quad-trees and R-trees) and c) exploration of query processing, concurrency control and transaction management for k-RP* files, and [89]. On the latter point, others have considered transactions to be inconsequential to the core problem of supporting more complex queries in P2P networks [72].

In architecting their secure wide-area Service Discovery Service (SDS), Hodes, Czerwinski et al. considered three possible designs for multi-criteria search – Centralization, Mapping and Flooding [90]. These correlate to the index classifications of Section 2 – Central, Distributed and Local. They discounted the centralized, Napster-like index for its risk of a single point of failure. They considered the hash-based mappings of Section 3 but concluded that it would not be possible to adequately partition data. A document satisfying many criteria would be wastefully stored in many partitions. They rejected full flooding for its lack of scalability. Instead, they devised a query filtering technique, reminiscent of Gnutella's query routing protocol (Section 4.1). Nodes push proactive summaries of their data rather than waiting for a query. Summaries are aggregated and stored throughout a server hierarchy, to guide subsequent queries. Some initial prototype measurements were provided for total load on the system, but not for load distribution. They put several issues forward for future work. The indexing needs to be flexible to change according to query and storage workloads. A mesh topology might improve on their hierarchic topology since query misses would not propagate to root servers. The choice is analogous to BGP meshes and DNS trees.

More recently, Cai, Frank et al. devised the Multi-Attribute Addressable Network (MAAN) [91]. They built on Chord to provide both multi-attribute and range queries, claiming to be the first to service both query types in a structured P2P system. Each MAAN node has O(log N) neighbours, where N is the number of nodes. MAAN multi-

attribute range queries require $O(\log N + N \times s_{\min})$ hops, where $s_{\min}$ is the minimum range selectivity across all attributes. Selectivity is the ratio of the query range to the entire identifier range. The paper assumed that a locality preserving hash function would ensure balanced load. Per Section 5.1, the arguments by Bharambe, Agrawal et al. have highlighted the shortcomings of this assumption [84]. MAAN required that the schema must be fixed and known in advance – adaptable schemas were recommended for subsequent attention. The authors also acknowledged that there is a selectivity breakpoint at which full flooding becomes more efficient than their scheme. This begs for a query resolution algorithm that adapts to the profile of queries. Cai and Frank followed up with RDFPeers [55]. They differentiate their work from other RDF proposals by a) guaranteeing to find query results if they exist and b) removing the requirement of prior definition of a fixed schema. They hashed <subject, predicate, object> triples onto the MAAN and reported routing hop metrics for their implementation. Load imbalance across nodes was reduced to less than one order of magnitude, but the specific measure was number of triples stored per node – skewed query loads were not considered. They plan to improve load balancing with the virtual servers of Section 5.1 [168].

5.3. Join Queries

Two research teams have done some initial work on P2P join operations. Harren, Hellerstein et al. initially described a three-layer architecture – storage, DHT and query processing. They implemented the join operation by modifying an existing Content Addressable Network (CAN) simulator, reporting “significant hot-spots in all dimensions: storage, processing and routing” [72]. They progressed their design more recently in the context of PIER, a distributed query engine based on CAN [22, 341]. They implemented two equi-join algorithms. In their design, a key is constructed from the “namespace” and the “resource ID”. There is a namespace for each relation and the resource ID is the primary key for base tuples in that relation. Queries are multicast to all nodes in the two namespaces (relations) to be joined. Their first algorithm is a DHT version of the symmetric hash join. Each node in the two namespaces finds the relevant tuples and hashes them to a new query namespace. The resource ID in the new namespace is the concatenation of join attributes. In the second algorithm, called “fetch matches”, one of the relations is already hashed on the join attributes. Each node in the second namespace finds tuples matching the query and retrieves the corresponding tuples from the the first relation. They leveraged two other techniques, namely the symmetric semi-join rewrite and the Bloom filter rewrite, to reduce the high bandwidth overheads of the symmetric hash join. For an overlay of 10,000 nodes, they simulated the delay to retrieve tuples and the aggregate

network bandwidth for these four schemes. The initial prototype was on a cluster of 64 PCs, but it has more recently been expanded to PlanetLab.

Triantafillou and Pitoura considered multicasting to large numbers of peers to be inefficient [76]. They therefore allocated a limited number of special peers, called range guards. The domain of the join attributes was divided, one partition per range guard. Join queries were sent only to range guards, where the query was executed. Efficient selection of range guards and a quantitive evaluation of their proposal were left for future work.

5.4. Aggregation Queries

Aggregation queries invariable rely on tree-structures to combine results from a large number of nodes. Examples of aggregation queries are Count, Sum, Maximum, Minimum, Average, Median and Top-K [92, 342, 343]. Figure 5 summarizes the tree and query characteristics that affect dependability.

The fundamental design choices for aggregation trees relate to how the overlay uses DHTs, how it repairs itself when there are failures, how many aggregation trees there are, and whether the tree is static or dynamic (Figure 5). Astrolabe is one of the most influential P2P designs included in Figure 5, yet it makes no use of DHTs [92]. Other designs make use of the internal trees of Plaxton-like DHTs. Others build independent tree structures on top of DHTs. Most of the designs repair the aggregation tree with periodic mechanisms similar to those used in the DHTs themselves. Willow is an exception [32]. It uses a Tree Maintenance Protocol to “zip” disjoint aggregation trees together when there are major failures. Yalagandula and Dahlin found reconfigurations at the aggregation layer to be costly, suggesting more research on techniques to reduce the cost and frequency of such reconfigurations [98]. Many of the designs use multiple aggregation trees, each rooted at the DHT node responsible for the aggregation attribute. On the other hand, the Self-Organized Metadata Overlay [56] uses a single tree and is vulnerable to a single point of failure at its root.

At the time of writing, researchers have just begun exploring the performance of queries in the presence of churn. Most designs are for best-effort queries. Bawa et al. devised a better consistency model, called Single-Site Validity [99] to qualify the accuracy of results when there is churn. Its price was a five-fold increase in the message load, when compared to an efficient but best-effort Spanning Tree. Gossip mechanisms are resilient to churn, but they delay aggregation results and incur high message cost for aggregation attributes with small read-to-write ratios.

6. Conclusions

Research on peer-to-peer networks can be divided into four categories – search, storage, security and applications. This critical survey has focused on search methods. While P2P networks have been classified by the

existence of an index (structured or unstructured) or the location of the index (local, centralized and distributed), this survey has shown that most have evolved to have some structure, whether it is indexes at superpeers or indexes defined by DHT algorithms. As for location, the distributed index is most common. The survey has characterized indexes as semantic and semantic-free. It has also critiqued P2P work on major query types. While much of it addresses work from 2000 or later, we have traced important building blocks from the 1990s.

The initial motivation in this survey was to answer the question, "How robust are P2P search networks?" The question is key to the deployment of P2P technology. Balakrishnan, Kaashoek et al. argued that the P2P architecture is appealing: the startup and growth barriers are low; they can aggregate enormous storage and processing resources; "the decentralized and distributed nature of P2P systems gives them the *potential* to be robust to faults or intentional attacks" [18]. If P2P is to be a disruptive technology in applications other than casual file sharing, then robustness needs to be practically verified [20].

The best comparative research on P2P dependability has been done in the context of Distributed Hash Tables (DHTs) [270]. The entire body of DHT research can be distilled to four main observations about dependability (Section 3.2). Firstly, static dependability comparisons show that no O(log N) DHT geometry is significantly more dependable than the other O(log N) geometries. Secondly, dynamic dependability comparisons show that DHT dependability is sensitive to the underlying topology maintenance algorithms (Figure 2). Thirdly, most DHTs use O(log N) geometries to suit ephemeral nodes, whereas the O(1) hop DHTs suit stable nodes – they deserve more research attention. Fourthly, although not yet a mature science, the study of DHT dependability is helped by recent simulation tools that support multiple DHTs [278].

We make the following four suggestions for future P2P research:

1) Complete the companion P2P surveys for storage, security and applications. A rough outline has been suggested in Figure 1, along with references. The need for such surveys was highlighted within the peer-to-peer research group of the Internet Research Task Force (IRTF) [17].

2) P2P indexes are maturing. P2P queries are embryonic. Work on more expressive queries over P2P indexes started to gain momentum in 2003, but remains fraught with efficiency and load issues.

3) Isolate the low-level mechanisms affecting robustness. There is limited value in comparing robustness of DHT geometries (like rings versus de Bruijn graphs), when robustness is highly sensitive to underlying topology maintenance algorithms (Figure 2).

4) Build consensus on robustness metrics and their acceptable ranges. This paper has teased out numerous measures that impinge on robustness, for example, the median query path length for a failure of x% of nodes, bisection width, path overlap, the number of alternatives available for the next hop, lookup latency, average live bandwidth (bytes/node/sec), successful routing rates, the number of timeouts (caused by a finger pointing to a departed node), lookup failure rates (caused by nodes that temporarily point to the wrong successor during churn) and clustering measures (edge expansion and node expansion). Application-level robustness metrics need to drive a consistent assessment of the underlying search mechanics.

7. Bibliography

[1] M. Roussopoulos, M. Baker, D. Rosenthal, T. Guili, P. Maniatis, and J. Mogul, 2 P2P of Not 2 P2P?, The 3rd Int'l Workshop on Peer-to-Peer Systems, February 26-27 2004.

[2] A. Rowstron and P. Druschel, Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems, IFIP/ACM Middleware 2001, Nov 2001.

[3] B. Yeager and B. Bhattacharjee, Peer-to-Peer Research Group Charter, http://www.irtf.org/charters/p2prg.html (2003)

[4] T. Klingberg and R. Manfredi, Gnutella 0.6, (2002)

[5] I. Clarke, A Distributed Decentralised Information Storage and Retrieval System, Undergraduate Thesis, 1999.

[6] B. Zhao, J. Kubiatowicz, and A. Joseph, Tapestry: an infrastructure for fault-tolerant wide-area location and routing, Report No. UCB/CSD-01-1141 2001.

[7] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, and H. Balakrishnan, Chord: A scalable peer-to-peer lookup service for internet applications, Proc. ACM SIGCOMM 2001 2001, pp. 149-160.

[8] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, A scalable content-addressable network, Proc. of the conf. on Applications, technologies, architectures and protocols for computer communications, August 27-31 2001, pp. 161-172.

[9] C. Tang, Z. Xu, and M. Mahalingam, pSearch: information retrieval in structured overlays, First Workshop on Hot Topics in Networks. Also Computer Communication Review, Volume 33, Number 1, January 2003, Oct 28-29 2002.

[10] W. Nejdl, S. Decker, and W. Siberski, Edutella Project, RDF-based Metadata Infrastructure for P2P Applications, http://edutella.jxta.org/ (2003)

[11] K. Aberer and M. Hauswirth, Peer-to-peer information systems: concepts and models, state-of-the-art, and future systems, ACM SIGSOFT Software Engineering Notes, Proc. 8th European software engineering conference held jointly with 9th ACM SIGSOFT international symposium on foundations of software engineering 26 (5) (2001)

[12] L. Zhou and R. van Renesse, P6P: a peer-to-peer approach to internet infrastructure, The 3rd Int'l Workshop on Peer-to-Peer Systems, February 26-27 2004.

[13] Citeseer, Citeseer Scientific Literature Digital Library, http://citeseer.ist.psu.edu/ (2004)

[14] D. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu, Peer-to-Peer Computing, HP Technical Report, HPL-2002-57 2002.

[15] K. Aberer and M. Hauswirth, An overview on peer-to-peer information systems, Workshop on Distributed Data and Structures WDAS-2002 2002.
[16] F. DePaoli and L. Mariani, Dependability in Peer-to-Peer Systems, IEEE Internet Computing 8 (4) (2004) 54-61.
[17] B. Yeager, Proposed research tracks, Email to the Internet Research Task Force IRTF P2P Research Group, Nov 10 2003.
[18] H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, Looking up data in P2P systems, Communications of the ACM 46 (2) (2003) 43-48.
[19] D. Kossmann, The state of the art in distributed query processing, ACM Computing Surveys 32 (4) (2000) 422-469.
[20] B. Gedik and L. Liu, Reliable peer-to-peer information monitoring through replication, Proc. 22nd Int'l Symp. on Reliable Distributed Systems, 6-8 Oct 2003, pp. 56-65.
[21] S.-M. Shi, Y. Guangwen, D. Wang, J. Yu, S. Qu, and M. Chen, Making peer-to-peer keyword searching feasible using multi-level partitioning, The 3rd Int'l Workshop on Peer-to-Peer Systems, February 26-27 2004.
[22] R. Huebsch, J. M. Hellerstein, N. Lanham, B. T. Loo, S. Shenker, and I. Stoica, Querying the Internet with PIER, Proc. 29th Int'l Conf. on Very Large Databases VLDB'03, September 2003.
[23] J. M. Hellerstein, Toward network data independence, ACM SIGMOD Record 32 (3) (2003) 34-40.
[24] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica, The impact of DHT routing geometry on resilience and proximity, Proc. 2003 conference on Applications, Technologies, Architectures and Protocols for Computer Communications 2003, pp. 381-394.
[25] N. Daswani, H. Garcia-Molina, and B. Yang, Open Problems in Data-sharing Peer-to-peer Systems, The 9th Int'l Conf. on Database Theory (ICDT 2003), Siena, Italy, 8-10 January (2003)
[26] B. Cooper and H. Garcia-Molina, Studying search networks with SIL, Second Int'l Workshop on Peer-to-Peer Systems IPTPS 03, 20-21 February 2003.
[27] M. Bawa, Q. Sun, P. Vinograd, B. Yang, B. Cooper, A. Crespo, N. Daswani, P. Ganesan, H. Garcia-Molina, S. Kamvar, S. Marti, and M. Schlossed, Peer-to-peer research at Stanford, ACM SIGMOD Record 32 (3) (2003) 23-28.
[28] B. Yang and H. Garcia-Molina, Improving search in peer-to-peer networks, Proc. 22nd IEEE Int'l Conf. on Distributed Computing Systems, July 2002.
[29] B. Yang and H. Garcia-Molina, Efficient search in peer-to-peer networks, Proc. 22nd Int'l Conf. on Distributed Computing Systems, July 2-5 2002.
[30] C. Plaxton, R. Rajaraman, and A. Richa, Accessing nearby copies of replicated objects in a distributed environment, ACM Symp. on Parallel Algorithms and Architectures (1997)
[31] B. Zhao, L. Huang, J. Stribling, S. Rhea, A. Joseph, and J. Kubiatowicz, Tapestry: A Resilient Global-Scale overlay for Service Deployment, IEEE Journal on Selected Areas in Communications 22 (1) (2004) 41-53.
[32] R. van Renesse and A. Bozdog, Willow: DHT, aggregation and publish/subscribe in one protocol, The 3rd Int'l Workshop on Peer-to-Peer Systems, February 26-27 2004.
[33] P. Ganesan, G. Krishna, and H. Garcia-Molina, Canon in G Major: Designing DHTs with Hierarchical Structure, Proc. Int'l Conf. on Distributed Computing Systems ICDCS 2004 2004.
[34] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, and H. Balakrishnan, Chord: a scalable peer-to-peer lookup protocol for Internet applications, IEEE/ACM Trans. on Networking 11 (1) (2003) 17-32.
[35] S. Rhea, T. Roscoe, and J. Kubiatowicz, Structured Peer-to-Peer Overlays Need Application-Driven Benchmarks, Proc. 2nd Int'l Workshop on Peer-to-Peer Systems IPTPS'03, February 20-21 2003.
[36] D. Loguinov, A. Kumar, and S. Ganesh, Graph-theoretic analysis of structured peer-to-peer systems: routing distances and fault resilience, Proc. 2003 conference on Applications, Technologies, Architectures and Protocols for Computer Communications, August 25-29 2003, pp. 395-406.
[37] F. Kaashoek and D. Karger, Koorde: A Simple Degree-optimal Hash Table, Second Int'l Workshop on Peer-to-Peer Systems IPTPS'03, 20-21 February 2003.
[38] N. Harvey, M. B. Jones, S. Saroiu, M. Theimer, and A. Wolman, SkipNet: A Scalable Overlay Network with Practical Locality Properties, Proc. Fourth USENIX Symp. on Internet Technologies and Systems USITS'03, March 2003.
[39] I. Gupta, K. Birman, P. Linga, A. Demers, and R. Van Renesse, Kelips: Building an efficient and stable P2P DHT through increased memory and background overhead, Second Int'l Workshop on Peer-to-Peer Systems IPTPS 03, Feb 20-21 2003.
[40] J. Cates, Robust and Efficient Data Management for a Distributed Hash Table, Master's Thesis, May 2003.
[41] J. Aspnes and G. Shah, Skip graphs, Proc. 14th annual ACM-SIAM symposium on discrete algorithms (2003) 384-393.
[42] K. Aberer, P. Cudre-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Punceva, and R. Schmidt, P-Grid: a self-organizing structured P2P system, ACM SIGMOD Record 32 (3) (2003) 29-33.

[43] B. Zhao, Y. Duan, L. Huang, A. Joseph, and J. Kubiatowicz, Brocade: landmark routing on overlay networks, First Int'l Workshop on Peer-to-Peer Systems IPTPS'02, March 2002.
[44] S. Ratnasamy, S. Shenker, and I. Stoica, Routing algorithms for DHTs: some open questions, Proc. First Int'l Workshop on Peer to Peer Systems, IPTPS 2002, March 2002.
[45] P. Maymounkov and D. Mazieres, Kademlia: A peer-to-peer information system based on the XOR metric, Proc. First Int'l Workshop on Peer to Peer Systems, IPTPS 2002, March 7-8 2002.
[46] D. Malkhi, M. Naor, and D. Ratajczak, Viceroy: a scalable and dynamic emulation of the butterfly, Proc. 21st annual symposium on principles of distributed computing PODC, July 21-24 2002, pp. 183-192.
[47] X. Li and C. Plaxton, On name resolution in peer to peer networks, Proc. ACM SIGACT Annual Workshop on Principles of Mobile Computing POMC'02 2002, pp. 82-89.
[48] N. Harvey, J. Dunagan, M. B. Jones, S. Saroiu, M. Theimer, and A. Wolman, SkipNet: A Scalable overlay Network with Practical Locality Properties, Microsoft Research Technical Report MSR-TR-2002-92 (2002)
[49] D. Karger, E. Lehman, T. Leighton, R. Panigraphy, M. Levin, and D. Lewin, Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web, ACM Symp. on Theory of Computing (1997)
[50] W. Litwin, M. Neimat, and D. Schneider, LH* - a scalable, distributed data structure, ACM Trans. on Database Systems (TODS) 21 (4) (1996) 480-525.
[51] R. Devine, Design and Implementation of DDH: A Distributed Dynamic Hashing Algorithm, Proc. 4th Int'l Conf. on Foundations of Data Organizations and Algorithms 1993.
[52] W. Litwin, M.-A. Niemat, and D. Schneider, LH* - Linear Hashing for Distributed Files, Proc. ACM Int'l Conf. on Management of Data SIGMOD, May 1993.
[53] C. Tempich, S. Staab, and A. Wranik, Remindin': semantic query routing in peer-to-peer networks, Proc. 13th conference on World Wide Web, New York, NY, USA, May 17-20 (2004) 640-649.
[54] B. T. Loo, R. Huebsch, I. Stoica, and J. M. Hellerstein, The case for a hybrid P2P search infrastructure, The 3rd Int'l Workshop on Peer-to-Peer Systems, February 26-27 2004.
[55] M. Cai and M. Frank, RDFPeers: a scalable distributed RDF repository based on a structured peer-to-peer network, Proc. 13th conference on World Wide Web, May 17-20 2004, pp. 650-657.
[56] Z. Zhang, S.-M. Shi, and J. Zhu, SOMO: Self-organized metadata overlay for resource management in P2P DHTs, Second Int'l Workshop on Peer-to-Peer Systems IPTPS'03, Feb 20-21 2003.
[57] B. Yang and H. Garcia-Molina, Designing a super-peer network, Proc. 19th Int'l Conf. on Data Engineering ICDE, March 2003.
[58] I. Tatarinov, P. Mork, Z. Ives, J. Madhavan, A. Halevy, D. Suciu, N. Dalvi, X. Dong, Y. Kadiyska, and G. Miklau, The Piazza peer data management project, ACM SIGMOD Record 32 (3) (2003) 47-52.
[59] W. Nejdl, W. Siberski, and M. Sintek, Design Issues and Challenges for RDF- and schema-based peer-to-peer systems, ACM SIGMOD Record 32 (3) (2003) 41-46.
[60] S. Joseph and T. Hoshiai, Decentralized Meta-Data Strategies: Effective Peer-to-Peer Search, IEICE Trans. Commun. E86-B (6 June) (2003) 1740-1753.
[61] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, Making gnutella-like P2P systems scalable, Proc. 2003 conference on Applications, Technologies, Architectures and Protocols for Computer Communications, August 25-29 2003, pp. 407-418.
[62] M. Bawa, G. S. Manku, and P. Raghavan, SETS: search enhanced by topic segmentation, Proc. 26th annual international ACM SIGIR conference on Research and Development in Information Retrieval 2003, pp. 306-313.
[63] H. Sunaga, M. Takemoto, and T. Iwata, Advanced peer to peer network platform for various services - SIONet Semantic Information Oriented Network, Proc. Second Int'l Conf. on Peer to Peer Computing, Sept 5-7 2002, pp. 169-170.
[64] M. Schlosser, M. Sintek, S. Decker, and W. Nejdl, HyperCuP - Hypercubes, Ontologies and P2P Networks, Springer Lecture Notes on Computer Science, Agents and Peer-to-Peer Systems Vol. 2530 (2002)
[65] M. Ripeanu, A. Iamnitchi, and P. Foster, Mapping the Gnutella network, IEEE Internet Computing 6 (1) (2002) 50-57.
[66] Q. Lv, S. Ratnasamy, and S. Shenker, Can Heterogeneity Make Gnutella Scalable?, Proc. 1st Int'l Workshop on Peer-to-Peer Systems IPTPS2002, March 7-8 2002.
[67] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, Search and replication in unstructured peer to peer networks, Proc. 16th international conference on supercomputing, June 22-26 2002, pp. 84-95.
[68] V. Kalogaraki, D. Gunopulos, and D. Zeinalipour-Yasti, XML schemas: integration and translation: A local search mechanism for peer to peer networks, Proc. 11th ACM international conference on Information and Knowledge management 2002, pp. 300-307.
[69] O. Babaoglu, H. Meling, and Montresor, Anthill: a framework for the development of agent-based peer-to-peer systems, Proc. IEEE Int'l Conf. on Distributed Computer systems 2002, pp. 15-22.
[70] M. Jovanovic, Modeling large-scale peer-to-peer networks and a case study of Gnutella, Master's Thesis 2001.

[71] I. Clarke, O. Sandberg, B. Wiley, and T. Hong, Freenet: A Distributed Anonymous Information Storage and Retrieval System. Springer, New York, USA, 2001.
[72] J. Harren, J. Hellerstein, R. Huebsch, B. Loo, S. Shenker, and I. Stoica, Complex queries in DHT-based peer-to-peer networks, Proc. First Int'l Workshop on Peer to Peer Systems IPTPS 2002, March 2002.
[73] B. Gedik and L. Liu, PeerCQ: A Decentralized and Self-Configuring Peer-to-Peer Information Monitoring System, Proc. 23rd Int'l Conf. on Distributed Computing Systems ICDCS2003, May 19-22 2003.
[74] B. T. Loo, R. Huebsch, J. M. Hellerstein, T. Roscoe, and I. Stoica, Analyzing P2P Overlays with Recursive Queries, Technical Report, CSD-04-1301, January 14 2004.
[75] R. Avnur and J. Hellerstein, Eddies: continuously adaptive query processing, Proc. 2000 ACM SIGMOD international conference on Management of Data 2000, pp. 261-272.
[76] P. Triantafillou and T. Pitoura, Towards a unifying framework for complex query processing over structured peer-to-peer data networks, Proc. First Int'l Workshop on Databases, Information Systems and Peer-to-Peer Computing DBISP2P, Sept 7-8 2003, pp. 169-183.
[77] A. Gupta, D. Agrawal, and A. E. Abbadi, Approximate range selection queries in peer-to-peer systems, Proc. First Biennial Conf. on Innovative Data Systems Research CIDR 2003 2003.
[78] S. Ratnasamy, P. Francis, and M. Handley, Range queries in DHTs, Technical Report IRB-TR-03-009, July 2003.
[79] S. Ramabhadran, S. Ratnasamy, J. Hellerstein, and S. Shenker, Brief announcement: prefix hash tree, Proc. 23rd Annual ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing, PODC 2004, July 25-28 2004, pp. 368-368.
[80] A. Andrzejak and Z. Xu, Scalable, efficient range queries for grid information services, Proc. Second IEEE Int'l Conf. on Peer to Peer Computing, September 2002.
[81] C. Schmidt and M. Parashar, Enabling flexible queries with guarantees in P2P systems, IEEE Internet Computing 8 (3) (2004) 19-26.
[82] E. Tanin, A. Harwood, and H. Samet, Indexing distributed complex data for complex queries, Proc. National Conf. on Digital Government Research 2004, pp. 81-90.
[83] P. Ganesan, M. Bawa, and H. Garcia-Molina, Online Balancing of Range-Partitioned Data with Applications to Peer-to-Peer Systems, Proc. 30th Int'l Conf. on Very Large Data Bases VLDB 2004, 29 August - 3 September 2004.
[84] A. Bharambe, M. Agrawal, and S. Seshan, Mercury: Supporting Scalable Multi-Attribute Range Queries, SIGCOMM'04, Aug 30-Sept 3 2004.
[85] K. Aberer, Scalable Data Access in P2P Systems Using Unbalanced Search Trees, Workshop on Distributed Data and Structures WDAS-2002 2002.
[86] K. Aberer, A. Datta, and M. Hauswirth, The Quest for Balancing Peer Load in Structured Peer-to-Peer Systems, Technical Report IC/2003/32 2003.
[87] W. Litwin, M.-A. Neimat, and D. Schneider, RP*: a family of order-preserving scalable distributed data structures, Proc. 20th Int'l Conf. on Very Large Data Bases VLDB'94, September 12-15 1994.
[88] M. Tsangou, S. Ndiaye, M. Seck, and W. Litwin, Range queries to scalable distributed data structure RP*, Proc. Fifth Workshop on Distributed Data and Structures, WDAS 2003, June 2003.
[89] W. Litwin and M.-A. Neimat, k-RP*s: a scalable distributed data structure for high-performance multi-attributed access, Proc. Fourth Int'l Conf. on Parallel and Distributed Information Systems (1996) 120-131.
[90] T. Hodes, S. Czerwinski, B. Zhao, A. Joseph, and R. Katz, An architecture for secure wide-area service discovery, Wireless Networks 8 (2/3) (2002) 213-230.
[91] M. Cai, M. Frank, J. Chen, and P. Szekely, MAAN: A Multi-Attribute Addressable Network for Grid Information Services, Proc. Int'l Workshop on Grid Computing, November 2003.
[92] R. van Renesse, K. P. Birman, and W. Vogels, Astrolabe: A robust and scalable technology for distribute system monitoring, management and data mining, ACM Trans. on Computer Systems 21 (2) (2003) 164-206.
[93] R. Bhagwan, G. Varghese, and G. Voelker, Cone: Augmenting DHTs to support distributed resource discovery, Technical Report, CS2003-0755, July 2003.
[94] K. Albrecht, R. Arnold, and R. Wattenhofer, Join and Leave in Peer-to-Peer Systems: The DASIS Approach, Technical Report 427, Department of Computer Science, November 2003.
[95] K. Albrecht, R. Arnold, and R. Wattenhofer, Aggregating Information in Peer-to-Peer Systems for Improved Join and Leave, Proc. Fourth IEEE Int'l Conf. on Peer-to-Peer Computing, 25-27 August 2004.
[96] A. Montresor, M. Jelasity, and O. Babaoglu, Robust aggregation protocol for large-scale overlay networks, Technical Report UBLCS-2003-16, December 2003.
[97] M. Jelasity, W. Kowalczyk, and M. van Steen, An Approach to Aggregation in Large and Fully Distributed Peer-to-Peer Overlay Networks, Proc. 12th Euromicro Conf. on Parallel, Distributted and Network based Processing PDP 2004, February 2004.
[98] P. Yalagandula and M. Dahlin, A scalable distributed information management system, SIGCOMM'04, Aug 30-Sept 3 2004.

[99] M. Bawa, A. Gionis, H. Garcia-Molina, and R. Motwani, The price of validity in dynamic networks, Proc. 2004 ACM SIGMOD Int'l Conf. on the management of data 2004, pp. 515-526.
[100] J. Aspnes, J. Kirsch, and A. Krishnamurthy, Load Balancing and Locality in Range-Queriable Data Structures, Proc. 23rd Annual ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing PODC 2004, July 25-28 2004.
[101] G. On, J. Schmitt, and R. Steinmetz, The effectiveness of realistic replication strategies on quality of availability for peer-to-peer systems, Proc. Third Int'l IEEE Conf. on Peer-to-Peer Computing, Sept 1-3 2003, pp. 57-64.
[102] D. Geels and J. Kubiatowicz, Replica management should be a game, Proc. SIGOPS European Workshop, September 2003.
[103] E. Cohen and S. Shenker, Replication strategies in unstructured peer to peer networks, Proc. 2002 conference on applications, technologies, architectures and protocols for computer communications 2002, pp. 177-190.
[104] E. Cohen and S. Shenker, P2P and multicast: replication strategies in unstructured peer to peer networks, Proc. 2002 conference on applications, technologies, architectures and protocols for computer communications 2002, pp. 177-190.
[105] H. Weatherspoon and J. Kubiatowicz, Erasure coding vs replication: a quantative comparison, Proc. First Int'l Workshop on Peer to Peer Systems IPTPS'02, March 2002.
[106] D. Lomet, Replicated indexes for distributed data, Proc. Fourth Int'l Conf. on Parallel and Distributed Information Systems, December 18-20 1996, pp. 108-119.
[107] V. Gopalakrishnan, B. Silaghi, B. Bhattacharjee, and P. Keleher, Adaptive Replication in Peer-to-Peer Systems, Proc. 24th Int'l Conf. on Distributed Computing Systems ICDCS 2004, March 23-26 2004.
[108] S.-D. Lin, Q. Lian, M. Chen, and Z. Zhang, A practical distributed mutual exclusion protocol in dynamic peer-to-peer systems, The 3rd Int'l Workshop on Peer-to-Peer Systems, February 26-27 2004.
[109] A. Adya, R. Wattenhofer, W. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. Douceur, J. Howell, J. Lorch, and M. Thiemer, Farsite: federated, available and reliable storage for an incompletely trusted environment, ACM SIGOPS Operating Systems Review, Special issue on Decentralized storage systems (2002) 1-14.
[110] A. Rowstron and P. Druschel, Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility, Proceedings ACM SOSP'01, October 2001, pp. 188-201.
[111] S. Rhea, C. Wells, P. Eaton, D. Geels, B. Zhao, H. Weatherspoon, and J. Kubiatowicz, Maintenance-Free Global Data Storage, IEEE Internet Computing 5 (5) (2001) 40-49.
[112] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, Oceanstore: An Architecture for global-scale persistent storage, Proc. Ninth Int'l Conf. on Architecture Support for Programming Languages and Operating Systems ASPLOS 2000, November 2000, pp. 190-201.
[113] K. Birman, The Surprising Power of Epidemic Communication, Springer-Verlag Heidelberg Lecture Notes in Computer Science Volume 2584/2003 (2003) 97-102.
[114] P. Costa, M. Migliavacca, G. P. Picco, and G. Cugola, Introducing reliability in content-based publish-subscribe through epidemic algorithms, Proc. 2nd international workshop on Distributed event-based systems 2003, pp. 1-8.
[115] P. Costa, M. Migliavacca, G. P. Picco, and G. Cugola, Epidemic Algorithms for Reliable Content-Based Publish-Subscribe: An Evaluation, The 24th Int'l Conf. on Distributed Computing Systems (ICDCS-2004), Mar 23-26, Tokyo University of Technology, Hachioji, Tokyo, Japan (2004)
[116] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, Epidemic Algorithms for Replicated Data Management, Proc. Sixth ACM Symp. on Principles of Distributed Computing 1987, pp. 1-12.
[117] P. Eugster, R. Guerraoiu, A. Kermarrec, and L. Massoulie, Epidemic information dissemination in distributed systems, IEEE Computer 37 (5) (2004) 60-67.
[118] W. Vogels, R. v. Renesse, and K. Birman, The power of epidemics: robust communication for large-scale distributed systems, ACM SIGCOMM Computer Communication Review 33 (1) (2003) 131-135.
[119] S. Voulgaris and M. van Steen, An epidemic protocol for managing routing tables in very large peer to peer networks, Proc. 14th IFIP/IEEE Workshop on Distributed Systems: Operations and Management, October 2003.
[120] I. Gupta, On the design of distributed protocols from differential equations, Proc. 23rd Annual ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing PODC 2004, July 25-28 2004, pp. 216-225.
[121] I. Gupta, K. P. Birman, and v. Renesse, Fighting Fire with Fire: Using randomized Gossip to combat stochastic scalability limits, Cornell University Dept of Computer Science Technical Report, March 2001.
[122] K. Birman and I. Gupta, Building Scalable Solutions to Distributed Computing Problems using Probabilistic Components, Submitted to the Int'l Conf. on Dependable Systems and Networks DSN-2004, Dependable Computing and Computing Symp. DCCS, June 28-July 1 2004.

[123] A. Ganesh, A.-M. Kermarrec, and L. Massoulie, Peer-to-peer membership management for gossip-based protocols, IEEE Trans. on Computers 52 (2) (2003) 139-149.
[124] N. Bailey, Epidemic Theory of Infectious Diseases and its Applications, Second Edition ed. Hafner Press, 1975.
[125] P. Eugster, R. Guerraoiu, S. Handurukande, P. Kouznetsov, and A.-M. Kermarrec, Lightweight Probabilistic Broadcast, ACM Trans. on Computer Systems 21 (4) (2003) 341-374.
[126] H. Weatherspoon and J. Kubiatowicz, Efficient heartbeats and repair of softstate in decentralized object location and routing systems, Proc. SIGOPS European Workshop, September 2002.
[127] G. Koloniari and E. Pitoura, Content-based Routing of Path Queries in Peer-to-Peer Systems, Proc. 9th Int'l Conf. on Extending DataBase Technology EDBT, March 14-18 2004.
[128] A. Mohan and V. Kalogaraki, Speculative routing and update propagation: a kundali centric approach, IEEE Int'l Conf. on Communications ICC'03, May 2002.
[129] G. Koloniari, Y. Petrakis, and E. Pitoura, Content-Based Overlay Networks for XML Peers Based on Multi-Level Bloom Filters, Proc. First Int'l Workshop on Databases, Information Systems and Peer-to-Peer Computing DBISP2P, Sept 7-8 2003, pp. 232-247.
[130] G. Koloniari and E. Pitoura, Bloom-Based Filters for Hierarchical Data, Proc. 5th Workshop on Distributed Data and Structures (WDAS) (2003)
[131] B. Bloom, Space/time trade-offs in hash coding with allowable errors, Communications of the ACM 13 (7) (1970) 422-426.
[132] M. Naor and U. Wieder, A Simple Fault Tolerant Distributed Hash Table, Second Int'l Workshop on Peer-to-Peer Systems (IPTPS 03), Berkeley, CA, USA, 20-21 February (2003)
[133] P. Maymounkov and D. Mazieres, Rateless codes and big downloads, Second Int'l Workshop on Peer-to-Peer Systems, IPTPS'03, February 20-21 2003.
[134] M. Krohn, M. Freedman, and D. Mazieres, On-the-fly verification of rateless erasure codes for efficient content distribution, Proc. IEEE Symp. on Security and Privacy, May 2004.
[135] J. Byers, J. Considine, M. Mitzenmacher, and S. Rost, Informed content delivery across adaptive overlay networks, Proc. 2002 conference on applications, technologies, architectures and protocols for computer communications 2002, pp. 47-60.
[136] J. Plank, S. Atchley, Y. Ding, and M. Beck, Algorithms for High Performance, Wide-Area Distributed File Downloads, Parallel Processing Letters 13 (2) (2003) 207-223.
[137] M. Castro, P. Rodrigues, and B. Liskov, BASE: Using abstraction to improve fault tolerance, ACM Trans. on Computer Systems 21 (3) (2003) 236-269.
[138] R. Rodrigues, B. Liskov, and L. Shrira, The design of a robust peer-to-peer system, 10th ACM SIGOPS European Workshop, Sep 2002.
[139] H. Weatherspoon, T. Moscovitz, and J. Kubiatowicz, Introspective failure analysis: avoiding correlated failures in peer-to-peer systems, Proc. Int'l Workshop on Reliable Peer-to-Peer Distributed Systems, Oct 2002.
[140] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, Vivaldi: A Decentralized Network Coordinate System, SIGCOMM'04, Aug 30-Sept 3 2004.
[141] E.-K. Lua, J. Crowcroft, and M. Pias, Highways: proximity clustering for massively scaleable peer-to-peer network routing, Proc. Fourth IEEE Int'l Conf. on Peer-to-Peer Computing, August 25-27 2004.
[142] F. Fessant, S. Handurukande, A.-M. Kermarrec, and L. Massoulie, Clustering in Peer-to-Peer File Sharing Workloads, The 3rd Int'l Workshop on Peer-to-Peer Systems, February 26-27 2004.
[143] T. S. E. Ng and H. Zhang, Predicting internet network distance with coordinates-based approaches, IEEE Infocom 2002, The 21st Annual Joint Conf. of the IEEE Computer and Communication Societies, June 23-27 2002.
[144] K. Hildrum, R. Krauthgamer, and J. Kubiatowicz, Object Location in Realistic Networks, Proc. Sixteenth ACM Symp. on Parallel Algorithms and Architectures (SPAA 2004), June 2004, pp. 25-35.
[145] P. Keleher, S. Bhattacharjee, and B. Silaghi, Are Virtualized Overlay Networks Too Much of a Good Thing?, First Int'l Workshop on Peer-to-Peer Systems IPTPS, March 2002.
[146] A. Mislove and P. Druschel, Providing administrative control and autonomy in structured peer-to-peer overlays, The 3rd Int'l Workshop on Peer-to-Peer Systems, June 9-12 2004.
[147] D. Karger and M. Ruhl, Diminished Chord: A Protocol for Heterogeneous SubGroup Formation in Peer-to-Peer Networks, The 3rd Int'l Workshop on Peer-to-Peer Systems, February 26-27 2004.
[148] B. Awerbuch and C. Scheideler, Consistent, order-preserving data management in distributed storage systems, Proc. Sixteenth ACM Symp. on Parallel Algorithms and Architectures SPAA 2004, June 27-30 2004, pp. 44-53.
[149] M. Freedman and D. Mazieres, Sloppy Hashing and Self-Organizing Clusters, Proc. 2nd Int'l Workshop on Peer-to-Peer Systems IPTPS '03, February 2003.

[150] F. Dabek, J. Li, E. Sit, J. Robertson, F. Kaashoek, and R. Morris, Designing a DHT for low latency and high throughput, Proc. First Symp. on Networked Systems Design and Implementation (NSDI'04), San Francisco, California, March 29-31 (2004) 85-98.
[151] M. Ruhl, Efficient algorithms for new computational models, Doctoral Dissertation, September 2003.
[152] K. Sollins, Designing for scale and differentiation, Proc. ACM SIGCOMM workshop on Future Directions in network architecture, August 25-27 2003.
[153] L. Massoulie, A. Kermarrec, and A. Ganesh, Network awareness and failure resilience in self-organizing overlay networks, Proc. 22nd Int'l Symp. on Reliable Distributed Systems, SRDS'03, Oct 6-8 2003, pp. 47-55.
[154] R. Cox, F. Dabek, F. Kaashoek, J. Li, and R. Morris, Practical,distributed network coordinates, ACM SIGCOMM Computer Communication Review 34 (1) (2004) 113-118.
[155] K. Hildrum, J. Kubiatowicz, S. Rao, and B. Zhao, Distributed object location in a dynamic network, Proc. 14th annual ACM symposium on parallel algorithms and architectures 2002, pp. 41-52.
[156] X. Zhang, Q. Zhang, G. Song, and W. Zhu, A Construction of Locality-Aware Overlay Network: mOverlay and its Performance, IEEE Journal on Selected Areas in Communications 22 (1) (2004) 18-28.
[157] N. Harvey, M. B. Jones, M. Theimer, and A. Wolman, Efficient recovery from organization disconnects in Skipnet, Second Int'l Workshop on Peer-to-Peer Systems IPTPS'03, Feb 20-21 2003.
[158] M. Pias, J. Crowcroft, S. Wilbur, T. Harris, and S. Bhatti, Lighthouses for scalable distributed location, Second Int'l Workshop on Peer-to-Peer Systems IPTPS'03, February 20-21 2003.
[159] K. Gummadi, S. Saroui, S. Gribble, and D. King, Estimating latency between arbitrary internet end hosts, Proc. SIGCOMM IMW 2002, November 2002.
[160] Y. Liu, X. Liu, L. Xiao, L. Ni, and X. Zhang, Location-aware topology matching in P2P systems, Proc. IEEE Infocomm, Mar 7-11 2004.
[161] G. S. Manku, Balanced binary trees for ID management and load balance in distributed hash tables, Proc. 23rd Annual ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing, PODC 2004, July 25-28 2004, pp. 197-205.
[162] J. Gao and P. Steenkiste, Design and Evaluation of a Distributed Scalable Content Delivery System, IEEE Journal on Selected Areas in Communications 22 (1) (2004) 54-66.
[163] X. Wang, Y. Zhang, X. Li, and D. Loguinov, On zone-balancing of peer-to-peer networks: analysis of random node join, Proc. joint international conference on measurement and modeling of computer systems, June 2004.
[164] D. Karger and M. Ruhl, Simple efficient load balancing algorithms for peer-to-peer systems, Proc. Sixteenth ACM Symp. on Parallel Algorithms and Architectures SPAA 2004, June 27-30 2004.
[165] D. Karger and M. Ruhl, Simple efficient load balancing algorithms for peer-to-peer systems, The 3rd Int'l Workshop on Peer-to-Peer Systems, February 26-27 2004.
[166] M. Adler, E. Halperin, R. Karp, and V. Vazirani, A stochastic process on the hypercube with applications to peer-to-peer networks, Proc. 35th ACM symposium on Theory of Computing 2003, pp. 575-584.
[167] C. Baquero and N. Lopes, Towards peer to peer content indexing, ACM SIGOPS Operating Systems Review 37 (4) (2003) 90-96.
[168] A. Rao, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, Load balancing in structured P2P systems, Proc. 2nd Int'l Workshop on Peer-to-Peer Systems, IPTPS'03, February 20-21 2003.
[169] J. Byers, J. Considine, and M. Mitzenmacher, Simple Load Balancing for Distributed Hash Tables, Second Int'l Workshop on Peer-to-Peer Systems IPTPS 03, 20-21 February 2003.
[170] P. Castro, J. Lee, and A. Misra, CLASH: A Protocol for Internet-Scale Utility-Oriented Distributed Computing, Proc. 24th Int'l Conf. on Distributed Computing Systems ICDCS 2004, March 23-26 2004.
[171] A. Stavrou, D. Rubenstein, and S. Sahu, A Lightwight, Robust P2P System to Handle Flash Crowds, IEEE Journal on Selected Areas in Communications 22 (1) (2004) 6-17.
[172] A. Selcuk, E. Uzun, and M. R. Pariente, A reputation-based trust management system for P2P networks, Fourth Int'l Workshop on Global and Peer-to-Peer Computing, April 20-21 2004.
[173] T. Papaioannou and G. Stamoulis, Effective use of reputation in peer-to-peer environments, Fourth Int'l Workshop on Global and Peer-to-Peer Computing, April 20-21 2004.
[174] M. Blaze, J. Feigenbaum, and J. Lacy, Trust and Reputation in P2P networks, http://www.neurogrid.net/twiki/bin/view/Main/ReputationAndTrust (2003)
[175] E. Damiani, D. C. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante, A reputation-based approach for choosing reliable resources in peer to peer networks, Proc. 9th conference on computer and communications security 2002, pp. 207-216.
[176] S. Marti, P. Ganesan, and H. Garcia-Molina, DHT routing using social links, The 3rd Int'l Workshop on Peer-to-Peer Systems, February 26-27 2004.
[177] G. Caronni and M. Waldvogel, Establishing trust in distributed storage providers, Proc. Third Int'l IEEE Conf. on Peer-to-Peer Computing, 1-3 Sept 2003, pp. 128-133.
[178] B. Sieka, A. Kshemkalyani, and M. Singhal, On the security of polling protocols in peer-to-peer systems, Proc. Fourth IEEE Int'l Conf. on Peer-to-Peer Computing, 25-27 August 2004.

[179] M. Feldman, K. Lai, I. Stoica, and J. Chuang, Robust Incentive Techniques for Peer-to-Peer Networks, ACM E-Commerce Conf. EC'04, May 2004.
[180] K. Anagnostakis and M. Greenwald, Exchange-based Incentive Mechanism for Peer-to-Peer File Sharing, Proc. 24th Int'l Conf. on Distributed Computing Systems ICDCS 2004, March 23-26 2004.
[181] J. Schneidman and D. Parkes, Rationality and self-Interest in peer to peer networks, Second Int'l Workshop on Peer-to-Peer Systems IPTPS'03, February 20-21 2003.
[182] C. Buragohain, D. Agrawal, and S. Subhash, A game theoretic framework for incentives in P2P systems, Proc. Third Int'l IEEE Conf. on Peer-to-Peer Computing, 1-3 Sept 2003, pp. 48-56.
[183] W. Josephson, E. Sirer, and F. Schneider, Peer-to-Peer Authentication with a Distributed Single Sign-On Service, The 3rd Int'l Workshop on Peer-to-Peer Systems, February 26-27 2004.
[184] A. Fiat and J. Saia, Censorship resistant peer to peer content addressable networks, Proc. 13th annual ACM-SIAM symposium on discrete algorithms 2002, pp. 94-103.
[185] N. Daswani and H. Garcia-Molina, Query-flood DoS attacks in gnutella, Proc. 9th ACM Conf. on Computer and Communications Security 2002, pp. 181-192.
[186] A. Singh and L. Liu, TrustMe: anonymous management of trust relationships in decentralized P2P systems, Proc. Third Int'l IEEE Conf. on Peer-to-Peer Computing, Sept 1-3 2003.
[187] A. Serjantov, Anonymizing censorship resistant systems, Proc. Second Int'l Conf. on Peer to Peer Computing, March 2002.
[188] S. Hazel and B. Wiley, Achord: A Variant of the Chord Lookup Service for Use in Censorship Resistant Peer-to-Peer Publishing Systems, Proc. Second Int'l Conf. on Peer to Peer Computing, March 2002.
[189] M. Freedman and R. Morris, Tarzan: a peer-to-peer anonymizing network layer, Proc. 9th ACM Conf. on Computer and Communications Security (2002) 193-206.
[190] M. Feldman, C. Papadimitriou, J. Chuang, and I. Stoica, Free-Riding and Whitewashing in Peer-to-Peer Systems, 3rd Annual Workshop on Economics and Information Security WEIS04, May 2004.
[191] L. Ramaswamy and L. Liu, FreeRiding: a new challenge for peer-to-peer file sharing systems, Proc. 2003 Hawaii Int'l Conf. on System Sciences, P2P Track, HICSS2003, January 6-9 2003.
[192] T.-W. Ngan, D. Wallach, and P. Druschel, Enforcing fair sharing of peer-to-peer resources, Second Int'l Workshop on Peer-to-Peer Systems, IPTPS'03, 20-21 February 2003.
[193] L. Cox and B. D. Noble, Samsara: honor among thieves in peer-to-peer storage, Proc. nineteenth ACM symposium on Operating System Principles 2003, pp. 120-132.
[194] M. Surridge and C. Upstill, Grid security: lessons for peer-to-peer systems, Proc. Third Int'l IEEE Conf. on Peer-to-Peer Computing, Sept 1-3 2003, pp. 2-6.
[195] E. Sit and R. Morris, Security considerations for peer-to-peer distributed hash tables, First Int'l Workshop on Peer-to-Peer Systems, March 2002.
[196] C. O'Donnel and V. Vaikuntanathan, Information leak in the Chord lookup protocol, Proc. Fourth IEEE Int'l Conf. on Peer-to-Peer Computing, 25-27 August 2004.
[197] K. Berket, A. Essiari, and A. Muratas, PKI-Based Security for Peer-to-Peer Information Sharing, Proc. Fourth IEEE Int'l Conf. on Peer-to-Peer Computing, 25-27 August 2004.
[198] B. Karp, S. Ratnasamy, S. Rhea, and S. Shenker, Spurring adoption of DHTs with OpenHash, a public DHT service, The 3rd Int'l Workshop on Peer-to-Peer Systems, February 26-27 2004.
[199] J. Considine, M. Walfish, and D. G. Andersen, A pragmatic approach to DHT adoption, Technical Report,, December 2003.
[200] G. Li, Peer to Peer Networks in Action, IEEE Internet Computing 6 (1) (2002) 37-39.
[201] A. Mislove, A. Post, C. Reis, P. Willmann, P. Druschel, D. Wallach, X. Bonnaire, P. Sens, J.-M. Busca, and L. Arantes-Bezerra, POST: A Secure, Resilient, Cooperative Messaging System, 9th Workshop on Hot Topics in Operating Systems, HotOS, May 2003.
[202] S. Saroiu, P. Gummadi, and S. Gribble, A measurement study of peer-to-peer file sharing systems, Proc. Multimedia Computing and Networking 2002 MMCN'02, January 2002.
[203] A. Muthitacharoen, R. Morris, T. Gil, and B. Chen, Ivy: a read/write peer-to-peer file system, ACM SIGOPS Operating Systems Review, Special issue on Decentralized storage systems, December 2002, pp. 31-44.
[204] A. Muthitacharoen, R. Morris, T. Gil, and B. Chen, A read/write peer-to-peer file system, Proc. 5th Symp. on Operating System Design and Implementation (OSDI 2002), Boston, MA, December (2002)
[205] F. Annexstein, K. Berman, M. Jovanovic, and K. Ponnavaikko, Indexing techniques for file sharing in scalable peer to peer networks, 11th IEEE Int'l Conf. on Computer Communications and Networks (2002) 10-15.
[206] G. Kan and Y. Faybishenko, Introduction to Gnougat, First Int'l Conf. on Peer-to-Peer Computing 2001 2001, pp. 4-12.
[207] R. Gold and D. Tidhar, Towards a content-based aggregation network, Proc. First Int'l Conf. on Peer to Peer Compuuting 2001, pp. 62-68.

[208] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, Wide-area cooperative storage with CFS, Proc. 18th ACM symposium on Operating System Principles 2001, pp. 202-215.
[209] M. Freedman, E. Freudenthal, and D. Mazieres, Democratizing Content Publication with Coral, Proc. First Symp. on Networked Systems Design and Implementation NSDI'04, March 29-31 2004, pp. 239-252.
[210] J. Li, B. T. Loo, J. Hellerstein, F. Kaashoek, D. Karger, and R. Morris, On the Feasibility of Peer-to-Peer Web Indexing and Search, Second Int'l Workshop on Peer-to-Peer Systems IPTPS 03, 20-21 February 2003.
[211] S. Iyer, A. Rowstron, and P. Druschel, Squirrel: a decentralized peer-to-peer web cache, Proc. 21st annual symposium on principles of distributed computing 2002, pp. 213-222.
[212] M. Bawa, R. Bayardo, S. Rajagopalan, and E. Shekita, Make it fresh, make it quick: searching a network of personal webservers, Proc. 12th international conference on World Wide Web 2003, pp. 577-586.
[213] B. T. Loo, S. Krishnamurthy, and O. Cooper, Distributed web crawling over DHTs, Technical Report, CSD-04-1305, February 9 2004.
[214] M. Junginger and Y. Lee, A self-organizing publish/subscribe middleware for dynamic peer-to-peer networks, IEEE Network 18 (1) (2004) 38-43.
[215] F. Cuenca-Acuna, C. Peery, R. Martin, and T. Nguyen, PlanetP: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities, Proc. 12th international symposium on High Performance Distributed Computing (HPDC), June 2002.
[216] M. Walfish, H. Balakrishnan, and S. Shenker, Untangling the web from DNS, Proc. First Symp. on Networked Systems Design and Implementation NSDI'04, March 29-31 2004, pp. 225-238.
[217] B. Awerbuch and C. Scheideler, Robust distributed name service, The 3rd Int'l Workshop on Peer-to-Peer Systems, February 26-27 2004.
[218] A. Iamnitchi, Resource Discovery in Large Resource-Sharing Environments, Doctoral Dissertation 2003.
[219] R. Cox, A. Muthitacharoen, and R. Morris, Serving DNS using a Peer-to-Peer Lookup Service, First Int'l Workshop on Peer-to-Peer Systems (IPTPS), March 2002.
[220] A. Chander, S. Dawson, P. Lincoln, and D. Stringer-Calvert, NEVRLATE: scalable resource discovery, Second IEEE/ACM Int'l Symp. on Cluster Computing and the Grid CCGRID2002 2002, pp. 56-65.
[221] M. Balazinska, H. Balakrishnan, and D. Karger, INS/Twine: A scalable Peer-to-Peer architecture for Intentional Resource Discovery, Proc. First Int'l Conf. on Pervasive Computing (IEEE) (2002)
[222] J. Kangasharju, K. Ross, and D. Turner, Secure and resilient peer-to-peer E-mail: design and implementation, Proc. Third Int'l IEEE Conf. on Peer-to-Peer Computing, 1-3 Sept 2003.
[223] V. Lo, D. Zappala, D. Zhou, Y. Liu, and S. Zhao, Cluster computing on the fly: P2P scheduling of idle cycles in the internet, The 3rd Int'l Workshop on Peer-to-Peer Systems, February 26-27 2004.
[224] A. Iamnitchi, I. Foster, and D. Nurmi, A peer-to-peer approach to resource discovery in grid environments, IEEE High Performance Distributed Computing 2002.
[225] I. Foster and A. Iamnitchi, On Death, Taxes and the Convergence of Peer-to-Peer and Grid Computing, Second Int'l Workshop on Peer-to-Peer Systems IPTPS 03, 20-21 February 2003.
[226] W. Hoschek, Peer-to-Peer Grid Databases for Web Service Discovery, Concurrency - Practice and Experience (2002) 1-7.
[227] K. Aberer, A. Datta, and M. Hauswirth, A decentralized public key infrastructure for customer-to-customer e-commerce, Int'l Journal of Business Process Integration and Management (2004)
[228] S. Ajmani, D. Clarke, C.-H. Moh, and S. Richman, ConChord: Cooperative SDSI Certificate Storage and Name Resolution, First Int'l Workshop on Peer-to-Peer Systems IPTPS, March 2002.
[229] J. Li, J. Stribling, T. Gil, R. Morris, and F. Kaashoek, Comparing the performance of distributed hash tables under churn, The 3rd Int'l Workshop on Peer-to-Peer Systems, February 26-27 2004.
[230] S. Shenker, The data-centric revolution in networking, Keynote Speech, 29th Int'l Conf. on Very Large Data Bases, September 9-12 2003.
[231] S. Gribble, A. Halevy, Z. Ives, M. Rodrig, and D. Suciu, What can databases do for P2P?, Proc. Fourth Int'l Workshop on Databases and the Web, WebDB2001, May 24-25 2001.
[232] D. Clark, The design philosophy of the DARPA internet protocols, ACM SIGCOMM Computer Communication Review, Symp. proceedings on communications architectures and protocols 18 (4) (1988)
[233] J.-C. Laprie, Dependable Computing and Fault Tolerance: Concepts and Terminology, Twenty-Fifth Int'l Symp. on Fault-Tolerant Computing, Highlights from Twenty-Five Years 1995, pp. 2-13.
[234] D. Clark, J. Wroclawski, K. Sollins, and R. Braden, Tussle in cyberspace: defining tomorrow's internet, Conf. on Applications, Technologies, Architectures and Protocols for Computer Communications 2002, pp. 347-356.
[235] Clip2, The Gnutella Protocol Specification, http://www.clip2.com (2000)
[236] Napster, http://www.napster.com (1999)
[237] J. Mishchke and B. Stiller, A methodology for the design of distributed search in P2P middleware, IEEE Network 18 (1) (2004) 30-37.

[238] J. Li and K. Sollins, Implementing aggregation and broadcast over distributed hash tables. Full report, http://krs.lcs.mit.edu/regions/docs.html (November) (2003)
[239] M. Castro, M. Costa, and A. Rowstron, Should we build Gnutella on a structured overlay?, ACM SIGCOMM Computer Communication Review 34 (1) (2004) 131-136.
[240] A. Singla and C. Rohrs, Ultrapeers: Another Step Towards Gnutella Scalability,, http://groups.yahoo.com/group/the_gdf/files/Proposals/Working%20Proposals/Ultrapeer/ Version 1.0, 26 November (2002)
[241] B. Cooper and H. Garcia-Molina, Ad hoc, Self-Supervising Peer-to-Peer Search Networks, Technical Report, http://www.cc.gatech.edu/~cooperb/odin/ 2003.
[242] R. Baeza-Yates and B. Ribeiro-Neto, Modern Information Retrieval. Addison Wesley, Essex, England, 1999.
[243] S. Sen and J. Wang, Analyzing peer-to-peer traffic across large networks, IEEE/ACM Trans. on Networking 12 (2) (2004) 219-232.
[244] H. Balakrishnan, S. Shenker, and M. Walfish, Semantic-Free Referencing in Linked Distributed Systems, Second Int'l Workshop on Peer-to-Peer Systems IPTPS 03, 20-21 February 2003.
[245] B. Yang, P. Vinograd, and H. Garcia-Molina, Evaluating GUESS and non-forwarding peer-to-peer search, The 24th Int'l Conf. on Distributed Computing Systems ICDCS'04, Mar 23-26 2004.
[246] A. Gupta, B. Liskov, and R. Rodrigues, One Hop Lookups for Peer-to-Peer Overlays, 9th Workshop on Hot Topics in Operating Systems (HotOS), 18-21 May 2003.
[247] A. Gupta, B. Liskov, and R. Rodrigues, Efficient Routing for Peer-to-Peer Overlays, First Symp. on Networked Systems Design and Implementation NSDI, March 2004.
[248] A. Mizrak, Y. Cheng, V. Kumar, and S. Savage, Structured superpeers: leveraging heterogeneity to provide constant-time lookup, IEEE Workshop on Internet Applications, June 23-24 2003.
[249] L. Adamic, R. Lukose, A. Puniyani, and B. Huberman, Search in power-law networks, Physical review E, The American Physical Society 64 (046135) (2001)
[250] F. Banaei-Kashani and C. Shahabi, Criticality-based analysis and design of unstructured peer-to-peer networks as "complex systems", Proc. 3rd IEEE/ACM Int'l Symp. on Cluster Computing and the Grid 2003, pp. 351-358.
[251] KaZaa, KaZaa Media Desktop, www.kazaa.com (2001)
[252] S. Sen and J. Wang, Analyzing peer-to-peer traffic across large networks, Proc. second ACM SIGCOMM workshop on Internet measurement, November 06-08 2002, pp. 137-150.
[253] DirectConnect, http:www.neo-modus.com (2001)
[254] S. Saroiu, K. Gummadi, R. Dunn, S. Gribble, and H. Levy, An analysis of Internet content delivery systems, ACM SIGOPS Operating Systems Review 36 (2002) 315-327.
[255] A. Loo, The Future or Peer-to-Peer Computing, Communications of the ACM 46 (9) (2003) 56-61.
[256] B. Yang and H. Garcia-Molina, Comparing Hybrid Peer-to-Peer Systems (extended), 27th Int'l Conf. on Very Large Data Bases, September 11-14 2001.
[257] D. Scholl, OpenNap Home Page, http://opennap.sourceforge.net/ (2001)
[258] S. Ghemawat, H. Gobioff, and S.-T. Leung, The Google file system, Proc. 19th ACM symposium on operating systems principles 2003, pp. 29-43.
[259] I. Clarke, S. Miller, T. Hong, O. Sandberg, and B. Wiley, Protecting Free Expression Online with Freenet, IEEE Internet Computing 6 (1) (2002)
[260] J. Mache, M. Gilbert, J. Guchereau, J. Lesh, F. Ramli, and M. Wilkinson, Request algorithms in Freenet-style peer-to-peer systems, Proc. Second IEEE Int'l Conf. on Peer to Peer Computing P2P'02, September 5-7 2002.
[261] C. Rohrs, Query Routing for the Gnutella Networks, http://www.limewire.com/developer/query_routing/keyword%20routing.htm Version 1.0 (2002)
[262] I. Clarke, Freenet's Next Generation Routing Protocol, http://freenetproject.org/index.php?page=ngrouting, 20th July 2003.
[263] A. Z. Kronfol, FASD: A fault-tolerant, adaptive scalable distributed search engine, Master's Thesis http://www.cs.princeton.edu/~akronfol/fasd/ 2002.
[264] S. Gribble, E. Brewer, J. M. Hellerstein, and D. Culler, Scalable, Distributed Data Structures for Internet Service Construction, Proc. 4th Symp. on Operating Systems Design and Implementation OSDI 2000, October 2000.
[265] K. Aberer, Efficient Search in Unbalanced, Randomized Peer-to-Peer Search Trees, EPFL Technical Report IC/2002/79 (2002)
[266] R. Honicky and E. Miller, A fast algorithm for online placement and reorganization of replicated data, Proc. 17th Int'l Parallel and Distributed Processing Symp., April 2003.
[267] G. S. Manku, Routing networks for distributed hash tables, Proc. 22nd annual ACM Symp. on Principles of Distributed Computing, PODC 2003, July 13-16 2003, pp. 133-142.
[268] S. Lei and A. Grama, Extended consistent hashing: a framework for distributed servers, Proc. 24th Int'l Conf. on Distributed Computing Systems ICDCS 2004, March 23-26 2004.

[269] W. Litwin, Re: Chord & LH*, Email to Ion Stoica, March 23 2004a.
[270] J. Li, J. Stribling, R. Morris, F. Kaashoek, and T. Gil, A performance vs. cost framework for evaluating DHT design tradeoffs under churn, Proc. IEEE Infocom, Mar 13-17 2005.
[271] S. Zhuang, D. Geels, I. Stoica, and R. Katz, On failure detection algorithms in overlay networks, Proc. IEEE Infocomm, Mar 13-17 2005.
[272] X. Li, J. Misra, and C. G. Plaxton, Active and Concurrent Topology Maintenance, The 18th Annual Conf. on Distributed Computing (DISC 2004), Trippenhuis, Amsterdam, the Netherlands, October 4 - October 7 (2004)
[273] K. Aberer, L. O. Alima, A. Ghodsi, S. Girdzijauskas, M. Hauswirth, and S. Haridi, The essence of P2P: a reference architecture for overlay networks, Proc. of the 5th international conference on peer-to-peer computing, Aug 31-Sep 2 2005.
[274] C. Tang, M. Buco, R. Chang, S. Dwarkadas, L. Luan, E. So, and C. Ward, Low traffic overlay networks with large routing tables, Proc. of the 2005 ACM Sigmetrics international conf. on Measurement and modeling of computer systems, Jun 6-10 2005, pp. 14-25.
[275] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, Handling churn in a DHT, Proc. of the USENIX Annual Technical Conference, June 2004.
[276] C. Blake and R. Rodrigues, High Availability, Scalable Storage, Dynamic Peer Networks: Pick Two, 9th Workshop on Hot Topics in Operating Systems (HotOS), Lihue, Hawaii, 18-21 May (2003)
[277] S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu, OpenDHT: a public DHT service and its uses, Proc. of the conf. on Applications, technologies, architectures and protocols for computer communications, Aug 22-26 2005, pp. 73-84.
[278] T. Gil, F. Kaashoek, J. Li, R. Morris, and J. Stribling, p2psim, a simulator for peer-to-peer protocols, http://www.pdos.lcs.mit.edu/p2psim/ (2003)
[279] K. Hildrum, J. D. Kubiatowicz, S. Rao, and B. Y. Zhao, Distributed object location in a dynamic network, Theory of Computing Systems (2004)
[280] N. Lynch, D. Malkhi, and D. Ratajczak, Atomic data access in distributed hash tables, Proc. Int'l Peer-to-Peer Symp., March 7-8 2002.
[281] S. Gilbert, N. Lynch, and A. Shvartsman, RAMBO II: Rapidly Reconfigurable Atomic Memory for Dynamic Networks, Technical Report, MIT-CSAIL-TR-890 2004.
[282] N. Lynch and I. Stoica, MultiChord: A resilient namespace management algorithm, Technical Memo MIT-LCS-TR-936 2004.
[283] J. Risson, K. Robinson, and T. Moors, Fault tolerant active rings for structured peer-to-peer overlays, Proc. of the 30th Annual IEEE Conf. on Local Computer Networks, Nov 15-17 2005, pp. 18-25.
[284] B. Awerbuch and C. Scheideler, Peer-to-peer systems for prefix search, Proc. 22nd annual ACM Symp. on Principles of Distributed Computing 2003, pp. 123-132.
[285] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica, Towards a common API for structured P2P overlays, Proc. Second Int'l Workshop on Peer to Peer Systems IPTPS 2003, February 2003.
[286] N. Feamster and H. Balakrishnan, Towards a logic for wide-area Internet routing, Proc. ACM SIGCOMM workshop on Future Directions in Network Architecture, August 25-27 2003, pp. 289-300.
[287] B. Ahlgren, M. Brunner, L. Eggert, R. Hancock, and S. Schmid, Invariants: a new design methodology for network architectures, Proc. ACM SIGCOMM workshop on Future Direction in Network Architecture, August 30 2004, pp. 65-70.
[288] R. Mahajan, M. Castro, and A. Rowstron, Controlling the cost of reliability in peer-to-peer overlays, Second Int'l Workshop on Peer-to-Peer Systems IPTPS'03, February 20-21 2003.
[289] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, Handling churn in a DHT, Report No. UCB/CSD-03-1299, University of California, also Proc. USENIX Annual Technical Conference, June 2003.
[290] M. Castro, M. Costa, and A. Rowstron, Performance and dependability of structured peer-to-peer overlays, Microsoft Research Technical Report MSR-TR-2003-94, December. Also 2004 Int'l Conf. on Dependable Systems and Networks, June 28-July 1 2003.
[291] D. Liben-Nowell, H. Balakrishnan, and D. Karger, Analysis of the evolution of peer-to-peer systems, Annual ACM Symp. on Principles of Distributed Computing 2002, pp. 233-242.
[292] L. Alima, S. El-Ansary, P. Brand, and S. Haridi, DKS(N,k,f): a family of low communication, scalable and fault-tolerant infrastructures for P2P applications, Proc. 3rd IEEE/ACM Int'l Symp. on Cluster Computing and the Grid (2003) 344-350.
[293] D. Karger and M. Ruhl, Finding nearest neighbours in growth-restricted metrics, Proc. 34th annual ACM symposium on Theory of computing 2002, pp. 741-750.
[294] S. Ratnasamy, A Scalable Content-Addressable Network, Doctoral Dissertation 2002.
[295] S. McCanne and S. Floyd, The LBNL/UCB Network Simulator.
[296] I. Abraham, D. Malkhi, and O. Dubzinski, LAND:Stretch (1+epsilon) Locality Aware Networks for DHTs, Proc. ACM-SIAM Symp. on Discrete Algorithms SODA-04 2004.

[297] M. Naor and U. Wieder, Novel architectures for P2P applications: the continuous-discrete approach, Proc. fifteenth annual ACM Symp. on Parallel Algorithms and Architectures, SPAA 2003, June 7-9 2003, pp. 50-59.
[298] N. D. de Bruijn, A combinatorial problem, Koninklijke Netherlands: Academe Van Wetenschappen 49 (1946) 758-764.
[299] J.-W. Mao, The Coloring and Routing Problems on de Bruijn Interconnection Networks, Doctoral Dissertation, July 18 2003.
[300] M. L. Schlumberger, De Bruijn communication networks, Doctoral Dissertation 1974.
[301] M. Imase and M. Itoh, Design to minimized diameter on build-block network, IEEE Trans. on Computers C-30 (6) (1981) 439-442.
[302] S. M. Reddy, D. K. Pradhan, and J. G. Kuhl, Direct graphs with minimal and maximal connectivity, Technical Report, School of Engineering, Oakland University (1980)
[303] R. A. Rowley and B. Bose, Fault-tolerant ring embedding in de Bruijn networks, IEEE Trans. on Computers 42 (12) (1993) 1480-1486.
[304] K. Y. Lee, G. Liu, and H. F. Jordan, Hierarchical networks for optical communications, Journal of Parallel and Distributed Computing 60 (2000) 1-16.
[305] M. Naor and U. Wieder, Know thy neighbor's neighbor: better routing for skip-graphs and small worlds, The 3rd Int'l Workshop on Peer-to-Peer Systems, February 26-27 2004.
[306] P. Fraigniaud and P. Gauron, The content-addressable networks D2B, Technical Report 1349, Laboratoire de Recherche en Informatique, January 2003.
[307] A. Datta, S. Girdzijauskas, and K. Aberer, On de Bruijn routing in distributed hash tables: there and back again, Proc. Fourth IEEE Int'l Conf. on Peer-to-Peer Computing, , 25-27 August 2004.
[308] W. Pugh, Skip lists: a probabilistic alternative to balanced trees, Proc. Workshop on Algorithms and Data Structures, August 17-19 1989, pp. 437-449.
[309] W. Pugh, Skip lists: a probabilistic alternative to balanced trees, Communications of the ACM 33 (6) (1990) 668-676.
[310] J. Gray, The transaction concept: Virtues and limitations, Proc. VLDB, September 1981.
[311] B. T. Loo, J. M. Hellerstein, R. Huebsch, S. Shenker, and I. Stoica, Enhancing P2P file-sharing with internet-scale query processor, Proc. 30th Int'l Conf. on Very Large Data Bases VLDB 2004, 29 August-3 September 2004.
[312] M. Stonebraker, P. Aoki, W. Litwin, A. Pfeffer, A. Sah, J. Sidell, C. Staelin, and A. Yu, Mariposa: a wide-area distributed database system, THE VLDB Journal - The Int'l Journal of Very Large Data Bases (5) (1996) 48-63.
[313] V. Cholvi, P. Felber, and E. Biersack, Efficient Search in Unstructured Peer-to-Peer Networks, Proc. Symp. on Parallel Algorithms and Architectures, July 2004.
[314] S. Daswani and A. Fisk, Gnutella UDP Extension for Scalable Searches (GUESS) v0.1, http://www.limewire.org/fisheye/viewrep/~raw,r=1.2/limecvs/core/guess_01.html (2002)
[315] A. Fisk, Gnutella Dynamic Query Protocol v0.1, Gnutella Developer Forum (2003)
[316] O. Gnawali, A Keyword Set Search System for Peer-to-Peer Networks, Master's Thesis 2002.
[317] Limewire, Limewire Host Count, http://www.limewire.com/english/content/netsize.shtml (2004)
[318] A. Fisk, Gnutella Ultrapeer Query Routing, http://groups.yahoo.com/group/the_gdf/files/Proposals/Working%20Proposals/search/Ultrapeer%20QRP/ v0.1 (2003)
[319] A. Fisk, Gnutella Dynamic Query Protocol, http://groups.yahoo.com/group/the_gdf/files/Proposals/Working%20Proposals/search/Dynamic%20Querying/ v0.1 (2003)
[320] S. Thadani, Meta Data searches on the Gnutella Network (addendum), http://www.limewire.com/developer/MetaProposal2.htm (2001)
[321] S. Thadani, Meta Information Searches on the Gnutella Networks, http://www.limewire.com/developer/metainfo_searches.html (2001)
[322] P. Reynolds and A. Vahdat, Efficient peer-to-peer keyword searching, ACM/IFP/USENIX Int'l Middleware Conference, Middleware 2003, June 16-20 2003.
[323] W. Terpstra, S. Behnel, L. Fiege, J. Kangasharju, and A. Buchmann, Bit Zipper Rendezvous, optimal data placement for general P2P queries, Proc. First Int'l Workshop on Peer-to-Peer Computing and Databases, March 14 2004.
[324] A. Singhal, Modern Information Retrieval: A Brief Overview, IEEE Data Engineering Bulletin 24 (4) (2001) 35-43.
[325] E. Cohen, A. Fiat, and H. Kaplan, Associative Search in Peer to Peer Networks: Harnessing Latent Semantics, IEEE Infocom 2003, The 22nd Annual Joint Conf. of the IEEE Computer and Communications Societies, March 30-April 3 2003.

[326] W. Muller and A. Henrich, Fast retrieval of high-dimensional feature vectors in P2P networks using compact peer data summaries, Proc. 5th ACM SIGMM international workshop on Multimedia Information Retrieval, November 7 2003, pp. 79-86.
[327] M. T. Ozsu and P. Valduriez, Principles of Distributed Database Systems, 2nd edition ed. Prentice Hall, 1999.
[328] G. Salton, A. Wong, and C. S. Yang, A vector space model for automatic indexing, Communications of the ACM 18 (11) (1975) 613-620.
[329] S. E. Robertson, S. Walker, and M. Beaulieu, Okapi at TREC-7: automatic ad hoc, filtering, VLC and filtering tracks, Proc. Seventh Text REtrieval Conference, TREC-7, NIST Special Publication 500-242, July 1999, pp. 253-264.
[330] A. Singhal, J. Choi, D. Hindle, D. Lewis, and F. Pereira, AT&T at TREC-7, Proc. Seventh Text REtrieval Conf. TREC-7, July 1999, pp. 253-264.
[331] K. Sankaralingam, S. Sethumadhavan, and J. Browne, Distributed Pagerank for P2P Systems, Proc. 12th international symposium on High Performance Distributed Computing HPDC, June 22-24 2003.
[332] I. Klampanos and J. Jose, An architecture for information retrieval over semi-collaborated peer-to-peer networks, Proc. 2004 ACM symposium on applied computing 2004, pp. 1078-1083.
[333] C. Tang, Z. Xu, and S. Dwarkadas, Peer-to-peer information retrieval using self-organizing semantic overlay networks, Proc. 2003 conference on Applications, Technologies, Architectures and Protocols for Computer Communications, August 25-29 2003, pp. 175-186.
[334] C. Tang and S. Dwarkadas, Hybrid global-local indexing for efficient peer-to-peer information retrieval, Proc. First Symp. on Networked Systems Design and Implementation NSDI'04, March 29-31 2004, pp. 211-224.
[335] G. W. Furnas, S. Deerwester, S. T. Dumais, T. K. Landauer, R. A. Harshman, L. A. Streeter, and K. E. Lochbaum, Information retrieval using a singular value decomposition model of latent semantic structure, Proc. 11th Annual Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval 1988, pp. 465-480.
[336] C. Tang, S. Dwarkadas, and Z. Xu, On scaling latent semantic indexing for large peer-to-peer systems, The 27th Annual Int'l ACM SIGIR Conf. SIGIR'04, ACM Special Interest Group on Information Retrieval, July 2004.
[337] W. Litwin and S. Sahri, Implementing SD-SQL Server: a Scalable Distributed Database System, CERIA Research Rerpot 2004-04-02, April 2004.
[338] M. Jarke and J. Koch, Query Optimization in Database Systems, ACM Computing Surveys 16 (2) (1984) 111-152.
[339] G. S. Manku, M. Bawa, and P. Raghavan, Symphony: Distributed Hashing in a Small World, Proc. 4th USENIX Symp. on Internet Technologies and Systems, March 26-28 2003.
[340] J. L. Bentley, Multidimensional binary search trees used for associative searching, Communications of the ACM 18 (9) (1975) 509-517.
[341] B. Chun, I. Stoica, J. Hellerstein, R. Huebsch, S. Jeffery, B. T. Loo, S. Mardanbeigi, T. Roscoe, S. Rhea, and S. Schenker, Querying at Internet Scale, Proc. 2004 ACM SIGMOD international conference on management of data, demonstration session 2004, pp. 935-936.
[342] P. Cao and Z. Wang, Efficient top-K query calculation in distributed networks, Proc. 23rd Annual ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing PODC 2004, July 25-28 2004, pp. 206-215.
[343] D. Psaltoulis, I. Kostoulas, I. Gupta, K. Birman, and A. Demers, Practical algorithms for size estimation in large and dynamic groups, Proc. Twenty-Third Annual ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing, PODC 2004, July 25-28 2004.
[344] R. van Renesse, The importance of aggregation, Springer-Verlag Lecture Notes in Computer Science "Future Directions in Distributed Computing". A. Schiper, A. A. Shvartsman, H. Weatherspoon, and B. Y. Zhao, editors. Springer-Verlag, Heidelberg volume 2584 (2003)

Figure 1

Taxonomy	Selected References
Search	[18, 21-29]
Semantic-Free Indexes *Plaxton Trees* *Rings* *Tori* *Butterflies* *de Bruijn Graphs* *Skip Graphs*	[2, 6, 7, 30-52]
Semantic Indexes *Keyword Lookup* *Peer Information Retrieval* *Peer Data Management*	[4, 53-71]
Search *Range Queries* *Multi-Attribute Queries* *Join Queries* *Aggregation Queries* *Continuous Queries* *Recursive Queries* *Adaptive Queries*	[20, 22, 23, 25, 32, 38, 41, 56, 72-100]
Storage	
Consistency & Replication *Eventual consistency* *Trade-offs…*	[101-112]
Distribution *Epidemics, Bloom Filters*	[39, 42, 90, 92, 113-131]
Fault Tolerance *Erasure Coding* *Byzantine Agreement …*	[40, 105, 132-139]
Locality	[24, 43, 47, 140-160]
Load Balancing	[37, 86, 100, 107, 151, 161-171]
Security	
Character *Identity* *Reputation and Trust* *Incentives*	[172-182]
Goals *Availability* *Authenticity* *Anonymity* *Access Control* *Fair Trading*	[25, 27, 71, 183-197]
Applications	[1, 198-200]
Memory *File Systems* *Web* *Content Delivery Networks* *Directories* *Service Discovery* *Publish / Subscribe ..*	[32, 90, 142, 201-222]
Intelligence *GRID* *Security…*	[223-228]

Figure 1 Classification of P2P Research Literature

Figure 2

CATEGORY	OPTIONS
NORMAL UPDATES	
Joins	Passive; active [272]
Leaves	Passive; active [272]
FAULT DETECTION	[271]
Maintenance	Proactive (periodic or keep-alive probes); reactive (correction-on-use, correction-on-failure) [273]
Report	Negative (all dead nodes, nodes recently failed); Positive (all live nodes; nodes recently recovered); [271]
TOPOLOGY SHARING	Yes; no (e.g., when a node detects a failed node, does it tell other nodes?) [271]
Multicast	Tree (explicit, implicit) [247, 274]
Gossip	Gossip timeouts; Number of contacts [39]
CORRECTIVE ACTION	
Routing	Rerouting actions (reroute once; route in parallel [270]; reject); Routing timeouts (TCP-style, virtual coordinates) [275]
Topology	Update action (evict/ replace/ tag node); update timeliness (immediate, periodic[275], delayed [276])

Figure 2 Topology Maintenance in Distributed Hash Tables

Figure 3

QUERY	DESCRIPTION
QUERY ROUTING	
Flooding	Peers only index local files so queries must propagate widely [4]
Policy-based	Choice of the next hop node: random; most/least recently used; most files shared; most results [245, 313]
Random walks	Parallel [67] or biased random walks [61, 66]
QUERY FORWARDING	
Iterative	Nodes perform iterative unicast searches of ultrapeers, until the desired number of results is achieved. See Gnutella UDP Extension for Scalable Searches (GUESS) [245, 314]
Recursive	
QUERY FLOW CONTROL	
Receiver-controlled	Receivers grant query tokens to senders, so as to avoid overload [61]
Reactive	Sender throttles queries when it notices receivers are discarding packets [61, 66]
Dynamic Time To Live	In the Dynamic Query Protocol, the sender adjusts the time-to-live on each iteration based on the number of results received, the number of connections left, and the number of nodes already theoretically reached by the search [315]
INDEX	**DESCRIPTION**
DISTRIBUTION	
Compression	Leaf nodes periodically send ultrapeers compressed query routing tables, as in the Query Routing Protocol [240]
One hop replication	Nodes maintain an index of content on their nearest neighbors [61, 313]
PARTITIONING	
By document	[210]
By keyword	Use an inverted list to find a matching document, either locally or at another peer [21]. Partition by keyword sets [316]
By document and keyword	Also known as Multi-Level Partitioning [21]
METRIC	**DESCRIPTION**
Query load	Queries per second per node/link [65, 245]
Degree (distribution)	The number of links per node [66, 313]. Early P2P networks approximated power-law networks, where the number of nodes with L links is proportional to L^{-k} where k is a constant [65]
Query delay	Reported in terms of time and hop count [61, 66]
Query success rate	The "Collapse Point" is the per-node query rate at which the query success rate drops below 90% [61]. See also [61, 245, 313].

Figure 3 Keyword Lookup in P2P Systems

Figure 4

Technique *(Underlying DHT [1])*	Reference
Peer-to-Peer (P2P)	
Locality Sensitive Hashing *(Chord)*	[77]
Prefix Hash Trees *(unspecified DHT)*	[78, 79]
Space Filling Curves *(CAN)*	[80]
Space Filling Curves *(Chord)*	[81]
Quadtrees *(Chord)*	[82]
Skip Graphs	[38, 41, 83, 100]
Mercury	[84]
P-Grid	[85, 86]
Scalable Distributed Data Structures (SDDS)	
RP*	[87, 88]

Note 1. Although several of the authors based their work on one particular DHT, it may be possible to port their work to others.

Figure 4 Solutions for Range Queries on P2P and SDDS indexes.

Figure 5

CLASSIFICATION	OPTIONS
Tree type	Doesn't use DHT [92], use internal DHT trees [95], use independent trees on top of DHTs
Tree repair	Periodic [93], exceptional [32]
Tree count	One per key, one per overlay [56]
Tree flexibility	Static [92], dynamic
Query interface	Install, update, probe [98]
Query distribution	Multicast [98], gossip [92]
Query applications	Leader election, voting, resource location, object placement and error recovery [98, 344]
Query semantics	
Consistency	Best-effort, eventual [92], snapshot / interval / single-site validity [99]
Timeliness	[344]
Lifetime	Continuous [97, 99], single-shot
No. attributes	Single, multiple
Query types	Count, sum, maximum, minimum, average, median, top k [92, 342, 343]

Figure 5 Aggregation Trees and Queries in P2P Networks

Key: Astrolabe [92]; Cone [93]; Distributed Approximative System Information Service (DASIS) [95]; Scalable Distributed Information Management System (SDIMS) [98]; Self-Organized Metadata Overlay (SOMO) [56]; Wildfire [99]; Willow [32]; Newscast [97]

*** Author Photo**

John Risson

Tim Moors

*** Author Biography**

John Risson has a decade of engineering experience with a leading telecommunications provider in Australia. He led the implementation team responsible for the introduction of national high-speed corporate data services. He also has two years of business development experience amongst internet service providers in north-east Asia with responsibilities for IP, MPLS and optical ethernet products.

He received his BEng (1st Class Hons) from the University of Queensland, Australia and the MEng from Monash University, Australia. He is a PhD candidate at the University of New South Wales, Australia. He is also a member of the ACM, the IEEE and the Australian Computer Society.

Tim Moors is a Senior Lecturer in the School of Electrical Engineering and Telecommunications at the University of New South Wales, in Sydney, Australia. He researches network reliability, transport protocols, and wireless LAN MAC protocols. Previously, he was with the Center for Advanced Technology in Telecommunications at Polytechnic University in New York, and prior to that, with the Communications Division of the Australian Defence Science and Technology Organisation. He received his PhD and BEng(Hons) degrees from universities in Western Australia (Curtin and UWA).