IEICE TRANS. COMMUN., VOL.E86–B, NO.6 JUNE 2003

INVITED PAPER *Special Issue on Content Delivery Networks*

Decentralized Meta-Data Strategies: Effective Peer-to-Peer Search

Sam JOSEPH[†], ***Nonmember and*** **Takashige HOSHIAI**[††a)], ***Regular Member***

SUMMARY Gnutella's service announcement in March 2000 stirred worldwide interest by referring to P2P model. Basically, the P2P model needs not the broker "the centralized management server" that until now has figured so importantly in prevailing business models, and offers a new approach that enables peers such as end terminals to discover out and locate other suitable peers on their own without going through an intermediary server. It seems clear that the wealth of content made available by peer-to-peer systems like Gnutella and Freenet have spurred many authors into considering how meta-data might be used to support more effective search in a distributed environment. This paper has reviewed a number of these systems and attempted to identify some common themes. At this time the major division between the different approaches is the use of a hash-based routing scheme.

key words: *P2P, meta-data, discovery, decentralized*

1. Introduction

Recent developments in peer-to-peer networks* have centered on the concept of distributed hashtables (DHT) [32] or content routing [26]. These approaches assume possession of a hash or other identifier that precisely specifies the document the user wishes to retrieve. Naturally there are some situations in which a user only has more general information about their needs, e.g. keywords or other meta-data. A number of different strategies for handling this kind of search in peer-to-peer networks have recently come to light. Several are summarized below along with an attempt to identify some common themes.

One could consider a document hash or other id to be a form of meta-data, but in this document we focus on more general meta-data strategies that involve either keywords or statements like those used in the Resource Description Format (RDF). Document hashes are distinguished from other meta-data in that they can be automatically generated from the document, and provide some degree of uniqueness. However hashes are difficult for humans to remember and the hash space does not generally reflect relationships in the document space; qualities that other types of meta-data try to provide.

Searching naturally divides into first determining the unique IDs of documents that match a query, then finding the actual documents with those IDs. Not all the systems outlined below support this division of labour. However, any system that maps meta-data to document IDs could clearly be used in tandem with one that maps those IDs to actual document locations. Once we have established the document location it remains to transfer it to the desired location, allowing us to divide the document retrieval process into three steps:

1. Meta-Data Search
2. Document Location Search
3. Document Download

Naturally the first step can be omitted, or merged with the 2nd, but superficially it would appear that separating the first two stages is the more efficient and flexible way to proceed. This division allows us the interesting possibility of creating "Mix & Match" document retrieval systems, particularly if inter-operability standards such as Tristero [36] catch on.

The rest of this paper will be structured as follows: In the next two subsections we review work in other fields that is related to our theme, before describing the different categories of meta-data. The remainder of the paper is broken up into four main sections. Section 2 reviews the strategies used by systems that operate with decentralized meta-data, while Sect. 3 reviews the systems themselves. Section 4 discusses some of the important issues facing systems that try to deal with decentralized meta-data, and Sect. 5 offers our conclusions.

1.1 Related Work

The field of P2P has much in common with multi-agent systems since they often require distributed search solutions (see Joseph & Kawamura [19]). However it is in the field of distributed content systems that we find approaches more directly related to the issues of decentralized meta-data. For example the Whois++ system

Manuscript received October 25, 2002.
Manuscript revised January 15, 2003.

[†]The author is with Strategic Software Division, The University of Tokyo, Tokyo, 113-8656 Japan.
[††]The author is with NTT Network Service Systems Laboratory, NTT Corporation, Musashino-shi, 180-8585 Japan.
a) E-mail: hoshiai.takashige@lab.ntt.co.jp

*Peer to Peer networks are large networks of similarly enabled nodes (server and client functionality) characterized by low reliability and high rates of churn; where churn refers nodes entering and leaving the network, connection failure etc.

of Deutsch et al. [13] provides a mechanism for forwarding queries to distributed servers on the basis of the content of those servers. The Harvest system of Bowman et al. [4] provided a similar service along with caching and replication, as did the Sheldon et al.'s Content Routing approach [31], which included query refinement and merging of result sets. Q-Pilot [33] is a more recent example that routes queries to different search engines based on their specialization.

These systems operate within a web-based environment where servers and clients are distinct and servers can be relied upon to have a static address and some degree of stability. The main difference with p2p systems is the melding of the client and server such that every user is much more likely to contribute content, which creates a greater demand for decentralized search. However the greater churn of the p2p network simultaneously makes meeting this demand all the more difficult.

Two exemplars of 1st generation Peer-to-Peer (P2P) systems are Gnutella (see Kan [17] for an overview) and Clarke et al.'s Freenet [8]. Freenet forwards queries according to beliefs about the contents of other nodes; considering file similarity in terms of closeness in a "key-space" generated by a cryptographic hash. Users must know a file's key in order to retrieve it from the network. Files are inserted into particular locations (as opposed to just shared in the Gnutella network) and combined with aggressive caching activity the arrangement of files ends up reflecting that of the key-space. Freenet search operates in serial, in contrast to the parallel broadcast search of Gnutella. A number of other systems have emerged that attempt to deal with the issues of distributed file storage. Chord [32] and CAN [26] provide distributed hashtable functionality, which allow the locations of files to be determined automatically from a single id, as well as offering guarantees about the number of messages required to retrieve a file, and the amount of state each node must store to support effective search. Tapestry [38] and Pastry [15] are variants of the Plaxton Mesh [25], using prefix/suffix address routing and publishing mechanisms based on this routing to distribute files. SWAN [3] improves on CAN's multi-dimensional routing by adding long-range connections that create a small world network [35].

Content Distribution/Delivery Networks (CDNs) redirect file requests to servers that can more rapidly respond to a particular client, and have a superficial similarity to the above P2P systems in that a file is requested via a unique identifier, and then returned from somewhere in the network. However most CDNs rely on centralized management schemes and in contrast to the above systems, use replication in order to improve response latency (see Krishnamurthy et al., [21] for a review). Very recently Chapeweske's Tornado project [7] has been attempting to merge P2P distributed functionality and CDN-like compatibility with existing HTTP servers and browsers.

1.2 Types of Meta-Data

Meta-data can be defined as additional data associated with some file or document. While the meta-data might appear within the document itself (such as a list of keywords), the idea is that the meta-data can be considered separately from the parent document. Taking an inclusive view of meta-data it seems that we can consider a number of possible types:

(1) **Document Hash**
An id generated from the document contents via some hashing algorithm that ideally will be unique to each document
(2) **Document Id**
An id assigned arbitrarily to a document according to some scheme—different from a hash in that it must be generated by some authority
(3) **Statistical representation**
A representation generated by performing a statistical operation on a document, that may involve statistics relating to a larger document collection, e.g. TFIDF (see Sect. 2.1.1)
(4) **Human assigned**
Keywords or more complex statements such as RDF (see Sect. 2.1.4)

The important differences between the types of meta-data are whether the representation is unique among documents, whether it can be automatically generated, and the logical complexity of the associations. It is conceivable that some future document summarization process might generate RDF statements automatically from a document body, but currently it would appear that these assignments require human intervention. Clearly the uniqueness property, or the ability to automatically generate meta-data has strong implications for how the meta-data can be used in a decentralized (or for that matter centralized) environment.

2. Strategies

In this section we review a number of different distributed meta-data strategies. Looking through the various systems we can see certain shared techniques that occur in various combinations. Table 1 gives an overview of which strategies are employed by which systems. The systems themselves will be considered in detail in Sect. 3.

The systems use overlapping sets of strategies; for example, FASD and PlanetP use TFIDF to rank documents, while both FASD and Anthill use hash based routing. In addition, Bloom filters are common to PlanetP, LimeWire Query Routing, YouServ

Table 1 Strategies overviews.

Edutella --- RDF Query hierarchy + JXTA
FASD --- Semantic routing + aggressive caching + TFIDF
Anthill --- Semantic routing + JXTA
Routing Indices --- Semantic routing + Bloom Filter + Document Number information
Alpine -- Group Formation + Peer Reputation Learning
Associative P2P Networks -- Semantic routing + Bayesian
PlanetP --- Semantic routing + Bloom Filter + TFIDF-related heuristic
Query Routing --- Semantic routing + Bloom Filter + Hops Information
SIONet --- XML + Semantic routing + Event place
JXTASearch -- XML QuerySpaces + Semantic routing
NeuroGrid -- Peer & RDF Reputation Learning + Semantic routing
Reptile -- Peer & XML Reputation Assignment
Semplesh -- RDF + Distributed Hash Table
HyperCuP -- QuerySpaces + HyperCube topology
YouServ – Bloom Filter + Registrar
pSearch – CAN + LSI + TFIDF

and Routing Indices, and the semantic routing concept is widely used. Alpine, JXTASearch and NeuroGrid share the strategy of learning node reputations, while JXTASearch, SIONet and Alpine all employ some form of query spaces.

The commonly used strategies and techniques are listed below, divided into meta-data markup approaches and more general techniques for filtering, routing and learning.

2.1 Approaches to Markup

(1) TFIDF

Term Frequency Inverse Document Frequency (TFIDF) is an Information Retrieval approach of Salton & Yang's [29] that rates the degree to which words are representative of a document. Thus, Term Frequency (TF), the number of times a word appears in the document, is adjusted by dividing by the Document Frequency (DF), the number of documents in which this word appears (assuming some collection of documents). Thus a high occurrence word like "the" is penalized for occurring in many documents, whereas a word like "simulation" that appears less frequently will receive a higher TFIDF rating (for the documents in which they occur). There are a number of variations on the basic TFIDF approach, and it can be calculated automatically from a document's text. TFIDF is a commonly used example of the Vector Space Model (VSM), whereby a document is represented as a vector to a point in a phase space. Alternative representations naturally include simple term frequency and LSI transformations (see below).

(2) Bayesian Analysis

A statistical procedure that tries to estimate a probability distribution based on an observed distribution. Probability estimates taken from the observed data are multiplied by a prior distribution (beliefs about what the probability distribution should be), and then normalized to create a posterior distribution. Thus both prior beliefs and observed data can be taken into account. Naturally Bayesian Analysis can be employed in a wide variety of fields, however it is mentioned here along with other markup schemes since it was used as the basis for one of the major alternatives to TFIDF in the information retrieval field. The Bayesian relevance weighting technique of Robertson and Sparck-Jones [27] involves attaching different weights to different search terms in order to reflect the way in which they are distributed over the document collection, and perhaps more importantly, the way that the search terms are distributed over relevant and irrelevant documents. Specifically, given some document collection, search terms and associated relevance judgements, each search term can be assigned a weight, which in turn allows the documents to be ranked in order of relevance to the query. The wide applicability of Bayesian Analysis is evidenced by its use in determining which nodes to query in the Associative P2P networks model.

(3) XML

eXtendible Markup Language—XML is a 'meta-language' (a language for describing other languages) that lets you design your own customized markup languages for limitless different types of documents. XML can do this because it's written in SGML, the international standard meta-language for text markup. XML is a simplified version of SGML, but it does allow the user to specify a DTD or Document Type Definition. The DTD describes the rules of a language's syntax. Given the DTD one can parse a document to check that it conforms to the language syntax.

A markup language (be it HTML or XML) allows you to include meta-statements about the text of the document and XML goes further allowing you to define a DTD that will specify the precise syntax of your meta-statements. So rather than just including a list of keywords in your document like so "keyword: apples" you can have "<keyword>apples</keyword>," which would specified by a DTD statement like "<!ELEMENT keyword (#PCDATA)>." An XML parser could then take the DTD and check that the document specified keywords in the correct format.

(4) RDF

Resource Description Framework—a framework for describing and interchanging metadata. RDF descriptions are typically exchanged in XML syntax; with the W3C RDF specification (http://www.w3.org/RDF/) defining a recommended encoding of RDF statements in XML. RDF statements consist of a resource, a property and a value, or Subject, Predicate, Object. A resource might be a URL or a document hash, and are named with URIs; e.g. traditional http URLs, or content hashes such as urn:sha1:blah-blah, so an example statement would be something like [urn:sha1:blahblahblah author "Dan"]. While the full RDF specification has many features regarding encoding of this data, essentially it is allowing us to go one step beyond pure resource-keyword association. If we think of RDF values as keywords we can see that use of RDF allows us to specify the type or relation between keyword and document, e.g. instead of just associat-

ing the term "Dan" with a document, we can say whether he is the author, or if the document is about him. Another important point is that all the elements in an RDF statement can be URIs, thus allowing us to distinguish between different definitions of the term Author, e.g. http://dc.org/predicate/author and http://ng.org/predicate/author. Dornfest and Brickley provide a good overview of RDF and other p2p meta-data issues [14].

(5) LSI

Latent Semantic Indexing (LSI) after Deerwester et al. [12] tries to overcome some of the problems faced by Vector Space Models (VSM) such as synonymy, polysemy, and errors in documents. LSI uses singular value decomposition (SVD) to transform and truncate a matrix of term vectors computed from a VSM like TFIDF to discover the semantics of terms and documents. For example: bird, eagle and wing are distinct terms but they might be considered similar under an appropriate LSI transformation. The axes of the new semantic subspace are computed using SVD, a technique for finding singular vectors and singular values of a matrix. Singular vectors are directions in the phase space, while singular values indicate the extent to which the variation in the data is explained by the corresponding singular vector. Thus if the terms bird, eagle and wing are frequently found in the same documents then they can be represented by a single semantic vector. Singular vectors with high singular values thus become a set of semantic vectors that can be normalized and used as the basis for a new VSM.

2.2 Techniques

(1) Bloom Filter

A Bloom filter is an approach to efficiently represent non-Boolean data in a bit-array. Bloom [6] introduced its use to improve performance for hyphenation/capitalization checking in word processing. The hash of each word is taken and the first N bits are taken to indicate the words position in a Boolean array. This bit can then be set to 1 or 0 to indicate something about that word; in Bloom's original application this indicated if the word could be hyphenated without resorting to complex rules. In peer-to-peer networks it can indicate the presence of a document that contains that word in a particular peer. Thus one can talk of a Bloom filter as summarizing a peer's contents. False positives can be created due to hash collisions, but not false negatives.

(2) Semantic Routing

Under semantic routing, a query is routed according to the meta-data contained in that query. Thus the use of semantic routing implies that each node will need to maintain some routing table that associates meta-data with other nodes. This data can be thought of as the "reputation" of other nodes from the perspective of the node storing the routing table.

(3) Reputation Learning

Reputation Learning is the process whereby belief in the ability of a peer to satisfy a query is updated in line with their previous responses to similar queries. Reputation learning differs from trust metrics in that trust metrics tend to focus on making an overall assessment of trust in a peer based on knowledge about the entire network. Reputation learning operates locally such that each peer maintains reputations information about only those other peers it has interacted with directly. The emphasis is on tracking a variety of types of trust in a small number of peers as opposed to tracking a single type of trust in all peers in the network.

(4) Query Spaces (Event Spaces, Group Formation)

Query spaces are subsets of nodes in a system that will be queried on a particular topic, or using a particular query format. JXTASearch uses queryspaces to specify the type of XML metadata that can be queried against for a given set of nodes. In SIONet the same concept is called EventSpaces, and in Alpine there are node groups associated with a particular topic. The idea behind a query or event space is to place some kind of limit on the set of nodes that will potentially receive a query. In all the above systems the nodes in a queryspace are not automatically queried. They provide a set from which the most suitable candidates will be selected.

(5) Trust Metrics

Trust metrics (see Levien [22] for a review) can be divided into scalar trust metrics and group trust metrics. In the simplest trust metric there are three inputs: a directed graph, a designated "seed" node indicating the root of trust, and a "target" node. In order to determine if the target node is trustworthy, we evaluate if the target is reachable from the seed. Each edge from a node s to a node t in the graph indicates that s believes that t is trustworthy. If no route can be found then there is no reason to believe that t is trustworthy, given the data available. More complex scalar trust metrics may associate weights with the graph edges, allowing a finer grained quantification, but analysis suggests them to be equally susceptible to attack. Group trust metrics like those used in Google [24] and Advogato [22] (both centralized systems) overcome some of the weaknesses of scalar trust metrics by performing random walks through the network starting from a number of trusted sources. Effectively the final trust in each node is equal to the probability that a random walk will end at that node. Thus it is important to make sure that the initial set of nodes are trustworthy, and that the random walk is not too short (some nodes will never be reached in a short walk), and not too long (the trustworthy influence of the initial nodes will end up having no effect, i.e. we might as well just select nodes at random). Effectively what Advogato and Google are doing is choosing some set of trustworthy nodes and saying

that trust in other nodes in the network is equal to the likelihood that a random walk will take us there. Thus highly connected nodes, and nodes closely connected to the trustworthy nodes will have higher rankings. Analysis suggests that these group trust metrics are less easy to attack, but there are many outstanding issues about how such a system could be deployed without a centralized hub (see Reptile discussion).

(6) Query Forwarding

In some systems each query is sent to only a single intended recipient. In others query forwarding is employed whereby a node unable to satisfy a query will pass it on to another node or nodes. Note that it is important to distinguish this from normal routing, where messages are forwarded from node to node until they reach a specific destination or destinations. A system that does not forward queries at the application level may well be relying on some type of message forwarding (routing) in the network layer.

(7) Distributed Hash Table (DHT)

A distributed hash table or DHT specifies a relation between entities (files, documents etc.) and a position in a distributed network. Chord [32] is an example of a distributed hashtable. Each entity is hashed and the position of the entity in the network is determined using the hash. Chord nodes take responsibility for a section of the hash space, and maintain a finger table that points to the next node in the hash space and then others at logarithmically increasing distances. Thus when a Chord node receives a query for an entity, the hash can be calculated and then the query is forwarded to the closest matching node in the finger table. Because of the logarithmic spacing of the finger table Chord offers guarantees that the desired entity can be located within $\mathrm{Log}(N)$ hops, where N is the total number of nodes in the network. Each node has detailed information about the nodes responsible for immediately following points in the hash space, and this information thins out as we move further away in the hash space. The logarithmic spacing ensures that each hop will at least halve the hash space distance to the desired entity.

(8) Caching

Decentralized meta-data involves the storage of meta-data information throughout a network. While some systems leave the meta-data created by a particular user in a particular location, others attempt to cache and replicate that data throughout the system. FASD employs the aggressive caching techniques of Freenet such that whenever meta-data information is passed through a node, it is cached in that location, and can be used to answer subsequent queries. Anthill InsertAnts carry individual document-keyword relations around the network storing them in multiple nests. The difference between this caching of meta-data and the storage of meta-data related routing information (such as semantic routing) is that it enables nodes that may initially have had no relation to the meta-data creator to respond directly to queries related to that meta-data without having to refer to the original source of the meta-data relations.

(9) CAN

The Content Addressable Network (CAN) [26] organizes the logical space of a distributed network as a d-dimensional Cartesian space (a d-torus) and partitions it into zones. Some number of nodes will take responsibility of each zone. Each file is represented by a point within the d-torus, and will be stored in the node or nodes that are responsible for the zone that contains that point. Routing involves transversing from one zone to another within the Cartesian space. Each node maintains connections to the nodes responsible for neighbouring zones allowing any other zone to be reached in the a number of hops proportional to the size and dimensionality of the space. As nodes leave and join the network zones merge and split appropriately to maintain the zone structure.

2.3 Strategy-System Summary

To summarize, Table 2 details the use of various common techniques for each of the systems. In the next section we consider the specifics of each system in turn.

3. System Summaries

In this section we consider each of the systems in term, with a brief summary of each. The summaries are not presented in any particular order, and for each system we emphasis the key distinguishing features and use of the various meta-data strategies and techniques.

3.1 Anthill

AntHill, created by Babaoglu et al. [1], is a complex adaptive systems analogy with lots of ant-related terminology. Part of the main thrust of the project is to eventually use genetic algorithms to evolve the ants, but work so far has focused on developing a live system and simulator that share code for efficiency. The project uses the JXTA framework and is designed as a general framework for P2P multi-agent based applications. Some preliminary simulation results are given for a document sharing application, employing a distributed keyword index. The crucial difference in their scheme (apart from calling messages ants) is that each nest (node) stores a routing table associating keyword hashes with sets of other nests. Different nests become associated with different parts of the hashed keyword space thus avoiding the problem that keyword space itself is highly clustered (presumably around various spellings of "Britney Spears"). Three types of ant (message) are used to support the document search. An insertAnt carries information about the document and its URL and is associated with a single keyword

Table 2 System-strategy summary.

	Markup-Scheme	Hash Usage	Bloom Filters	Semantic Routing	Feedback Learning	Query Forwarding
Edutella	RDF	No	No	Parallel	No	Yes
FASD	Keyword - TFIDF	Yes	No	Serial	No	Yes
Anthill	Keyword	Yes	No	Parallel	No	Yes
Routing Indices	Keyword/Topic	No	Yes	Serial	No	Yes
Alpine	Keyword [plug in modules]	No	No	Parallel	Yes	No
Associative P2P Networks	Keyword/Topic	No	No	No	No	Yes
PlanetP	Keyword - TFIDF	No	Yes	Serial	No	No
Query Routing	Keyword	No	Yes	Parallel	No	Yes
SIONet	XML	Yes	No	Parallel	Yes	Yes
JXTASearch (JXTA)	XML	No	No	Parallel	Yes	Yes
NeuroGrid	RDF	No	No	Parallel	Yes	Yes
Reptile	XML	No	No	No	Trust Metrics	No
Semplesh	RDF	Yes	No	Serial	No	No
HyperCuP	Keyword	No	No	Separate HyperCubes	No	Yes
Usearch (YouServ)	Keyword	No	Yes	Parallel	Yes	No?
pSearch	TFIDF/LSI	Yes	No	No	IR Query Expansion	No

hash. The insertAnt wanders around adding the document hash and URL to each nest, updating the hashed keyword—nest table and then moving to other nodes that store data about similar keywords. One insertAnt is generated for each keyword that is to be associated with the new document. A searchAnt is generated for each keyword in a query, and moves around generating replyAnts at each nest that has a related document. Both the insertAnt and searchAnt are forwarded to other nests using routing tables that associate keyword hashes with sets of other nests. The replyAnt carries the relevant URL back to the original nest, and the searchAnt continues searching until its TTL expires. Thus a set of URLs related to the query keywords is sent back to the original query location. Initial simulations show the path length dropping over time, even with hard limits set on the sizes of routing tables.

3.2 FASD

Kronfol's FASD [18] (fault-tolerant, adaptive, scalable, and distributed) is a keyword search layer for Freenet. The idea is that when documents are inserted into Freenet the TFIDF values for the words in the document are calculated and used to create a vector space representation of the document. This is a meta-data key that indicates how representative each word in the document is and a pointer to the document itself. Thus the meta-data key is inserted into Freenet at the same time as the document, and the meta-data key ends up in a different location from the document. Freenet nodes store routing tables that associate keys with other nodes. In normal Freenet operation these keys are document hashes, but FASD uses meta-data keys (i.e. TFIDF vectors). A closeness operator is required in order to know where to route an insert request or a search query. In Freenet, document searches use simple distance in the hash space. FASD uses cosine correlation to assess doc-doc similarity and doc-query similarity. Thus a serial search is performed in which each node tries to pass the query to a node that holds meta-data keys that are similar to the query. Each node tries to pass back the top n metadata keys that are found in the serial search starting from itself. The current set of meta-data scores (degrees of closeness) are passed along with the search query, allowing each node to ensure it only passes back higher quality matches. Ultimately the originally querying node will receive a set of meta-data keys, which can be used to retrieve the documents themselves. Kronfol performs simulations showing that use of the closeness operator is much more effective than random forwarding, and that FASD exhibits similar scaling characteristics to Freenet.

3.3 Edutella

Nejdl et al.'s Edutella [23] project is developing a fully featured RDF query language that will allow a variety

of RDF queries to be supported on top of the JXTA framework. The main focus appears to be on developing a query model with multiple layers, the most detailed of which will allow complex recursive queries. The routing of queries is supported through a peer registration framework, whereby each peer registers with a local hub its data schema and the level of query complexity it will support. An initial Edutella application is being created with educational materials in mind, i.e. documents that require structured meta-data. As such the project falls more into the distributed database systems category, rather than the high-churn, low reliability environment often associated with P2P systems like Gnutella and Freenet.

3.4 Routing Indices

Crespo & Garcia-Molina [10] have each node use a local routing index to choose which neighbouring nodes to forward queries to. These Routing Indices (RIs) specify relations between query "topics" and neighbouring nodes, meaning that in their basic "compound" form, the indices are proportional in size to the number of neighbours and topics rather than the number of documents in the system. Two variations are presented, namely "hop-count" and "exponential" indices, that incorporate information about how many hops along a particular route content is likely to be discovered. Simulations based on serial routing indicate that RIs can improve performance over flooding by 1 or 2 orders of magnitude, and 50–100% over random routing.

RIs are created and maintained by each node aggregating and sending their indices to their neighbors. Thus any RI update leads to a cascade of updates as each node passes on its new RI. Similarly when a node leaves the network, each neighbor node must remove entries, recompute RIs and forward the change and so on. Crespo & Garcia-Molina suggest there is a trade off between the advantages gained with RIs and the overhead generated in terms of updates. Numbers taken from Gnutella suggest that it could benefit from this approach.

3.5 LimeWire Query Routing

Rohr's Query Routing [28] involves each node creates a routing table by hashing the keywords from the files it stores, and then swapping these tables with neighbours. Inspired by an idea of Prinkey's that used an efficient bitmap encoding to reduce the size of the routing tables being exchanged between nodes. Query Routing is an extension that overcomes Prinkey's assumption of a tree network.

Prinkey's bitmap scheme involved hashing each keyword and storing the results in a bitmap. For example, if "good" hashes to 2 and "book" to 5, the routing table for {"good," "book"} can be compressed to the bitmap 001001.... The tables formed through this process are an example of Bloom filters. Rohr's major modification is to the representation of the keywords. He replaces the binary code with an integer code, each keyword being indicated not just as present or absent, but by an integer that tells us how many hops away a match to this keyword can be found. As routing tables are propagated from node to node the integers can be incremented allowing unreachable matches to be pruned off the routing table by setting them to some arbitrarily high value. High occurrences of this "infinity" value can then aid compression, e.g. a scarce encoding will be created.

3.6 Associative Peer to Peer Networks

Associative Peer-to-Peer Networks involve the use of possession rules stored in each node, that describe the precise route to other nodes that contain particular content. This allows the routing of query messages to more closely match the underlying network topology. Cohen et al. [9] perform simulations to compare a number of strategies. Random search, search biased to nodes with more rules (i.e. more content in general), search biased towards nodes with rules relating to content being search for (RAPIER) and a Greedy Approximation Strategy (GAS) that biases search towards query terms with higher information (i.e. those that are less frequent across the network). The GAS strategy uses a Bayesian approach that also allows information from failed searches to be used to update the search priorities (i.e. the posterior probability can be adjusted as certain nodes are eliminated from enquiries). Serial Simulations indicated that the best strategy depended on how rare an item was being searched for, but GAS and RAPIER consistently outperformed the simpler strategies. It remains unclear how much effort must be expended to maintain up-to-date associative rules throughout the network.

3.7 Alpine

Alpine nodes allow the user to define any number of groups, e.g. a "music" group, a "photography" group, and add other nodes to those groups. Thus a group of friends could simply create a group within which they wanted to share certain files (by adding each other to their local groups). Note that group information is local to each node, so that each node may use a different term to refer to the group, and the group may contain slightly different members for each local user.

Each node maintains an index of its local content. Queries for content are passed between members of a group and each node assigns a quality rating to the other members of the group depending on how well they can respond to the queries. This allows the distribution of queries within this group to be optimized,

i.e. queries can be sent to those peers most likely to be able to answer them. New members may be introduced into the group via transitive introduction, or an existing member may add them explicitly. A user may form a new group with an existing set of nodes by cloning the "default" group that contains all the nodes currently known about by the local node.

The process of ranking the other nodes in a group may use implicit information such as uptime, and also explicit information such as the user indicating that they got spammed. Each node's ranking within the group is aggregated over all queries. The relationship between a group and the kinds of queries generated is not fixed. One of the main features of Alpine is that there is no query forwarding—each node maintains connections to all members of a particular group using lightweight UDP connections.

3.8 PlanetP

Cuenca-Acuna et al. [11] adapt the TFIDF strategy to work with summaries of document indexes, rather than individual document data. The index of content of each peer is summarized using a Bloom filter. The summaries for each peer are diffused through the whole network using gossiping algorithms. The collection of Bloom filters allows each peer to approximate the inverted index of the entire community. In terms of choosing which peers to query first, nodes are ranked according to the presence of the query term in their Bloom filters, adjusted to the extent that the term is capable of resolving between different peers, i.e. the IDF term in TFIDF gives us an indication of how a term is used across a document collection, Cuenca-Acuna et al. use a similar approach to determine how a term is distributed across a set of peers. Thus a query for "classical mp3" would involve a calculation over the Bloom filters to determine that while most nodes had "mp3" related documents, "classical" related documents occurred in only a few nodes—thus these nodes would receive a higher ranking and be preferentially queried. Nodes are queried in series and the query process is halted after some significant number of nodes has failed to contribute documents to the top k documents. The precise details of the stopping condition are determined by a heuristic that is a function of the number of nodes in the network, and the value k. Simulations indicate that for networks of several thousand nodes the recall and precision of a centralized search engine can be matched, although naturally the time and bandwidth taken to diffuse the Bloom filters to all nodes scales as a linear function of the nodes in the network.

3.9 SIONet

Hoshiai's SIONet [16] is a "brokerless" meta network that, in contrast to conventional networks that require a destination address for packets to reach their proper destinations, delivers packets based on semantic information. This enables entities to perform search and discovery within a distributed environment. There are a number of different types of SIONet configuration elements—semantic switches, routers, and gateways as well as event-places and sessions that work together as needed to build the self-network organization from the bottom up. SIONet's central concepts are filter based subscription and event delivery. Individual entities describe the kinds of event they would like to receive using XML filters. Network entities then forward matching events to the appropriate entities.

The event place concept is important and allows the network to restrict certain classes of queries to certain parts of the network, to realize 'self-organization.'

3.10 JXTA Search and JXTA

JXTA is short for Juxtapose—a peer-to-peer interoperability framework created by Sun Microsystems. It incorporates a number of protocols, but the most relevant to our discussion of decentralized meta-data is the Peer Discovery Protocol (PDP). PDP allows a peer to advertise its own resources, and discover the resources from other peers. Every peer resource is described and published using an advertisement, which is an XML document that describes a network resource. JXTASearch operates over the lower level JXTA protocols.

JXTASearch is based on an earlier project called "InfraSearch." InfraSearch was based on the idea of distributing queries to network peers best capable of answering them. JXTASearch is a part of Sun's JXTA project and is comprised of Hubs, Consumers and Providers. The Hub concept is similar to that used in the Gnutella Reflector and FastTrack systems. JXTASearch specifies an XML Query Routing Protocol (QRP) that allows information providers to register available services at search hubs. Search queries from the consumers are received by the hubs and then forwarded to the most appropriate providers based on how well the query matches the provider registration. Information providers register for the queries they are interested in receiving by specifying a "queryspace" and a set of XML predicates that describe possible queries. These predicates are sets of clauses in conjunctive normal form (sets of OR possibilities ANDed together), where individual clauses can refer to any single element of the XML that is permitted in the queryspace. Thus a set of predicates and a "queryspace" definition comprise the information provider registration, which is then added to an inverted index for that particular hub (one index for each queryspace), each clause becoming a pattern/posting pair (or XML path—provider association).

When an XML query arrives at the search hub, the index is queried against each element of the in-

coming query to generate a list of possible information providers. Forwarding depends on how many of an information provider's registered patterns match the query. The possibility of feedback-based scoring of the individual pattern/posting elements is also mentioned, however the effects of different feedback schemes, or indeed different matching threshold possibilities are not explored. QuerySpaces allow the system to support multiple different query domains, each with their own specific XML syntax. Individual providers can register to receive specific or general queries, for example, all queries for jpeg files, or all queries for books with the word "java" in the title. The hub collects all queries returned by the providers, merges them and passes them back to the original querant.

3.11 NeuroGrid

Joseph's NeuroGrid [20] maintains routing tables at each node that associate nodes with query keywords. The strength of these associations is varied over time in response to user feedback. For example a user may generate a query for "used car auctions," which will be forwarded by the local node to other nodes that have previously been associated with these keywords. As the user receives potential matches from other nodes the local node monitors whether the user ignores them, or performs some feedback activity, either implicit—bookmarking the match, or explicit—clicking a "spam" button. Depending on this feedback the local node adjusts the relation between the query keywords and the remote node that provided the recommendation. Thus nodes that consistently provide results unsatisfactory to users will not be queried in future. Each node forwards incoming queries to a subset of the most relevant nodes, and simulations show that this allows the path length to drop to close to 1 in a 1000 node network, where a random forwarding case fails to show much improvement over starting conditions. The current simulations assume an "ideal" user and document model, where meta-data is accurate and users get what they desire. A web-based version of the system has been implemented that takes real user feedback into account, and supports full RDF metadata markup.

3.12 Reptile

Reptile, designed by Burton [5], is a distributed newspaper that uses reputation to build a distributed publication, subscription and search infrastructure. Reptile uses reputation to determine which peers to route article queries to (e.g. give me all articles generated in the last 10 minutes). Reputation determines channel, articles and user rankings. For example, the average Reptile peer receives about 100–300 articles a day syndicated from other peers, weblogs, and web sites. Using reputation Reptile narrows this down to the highest quality articles in a manageable format; top 10 articles for the day, top 100 articles for the month, top 10 articles from an individual peer, etc. Reputations networks are composed of many individual "certifications" which are signed chunks of meta-data indicating the degree of trust placed in an entity along with a confidence rating. Centralized systems can transitively calculate the trustworthiness of any one individual entity by taking all reputes into account. In the decentralized environment, one must calculate trust based on locally stored reputes, or query the network to gain enough reputes to make a trust judgement. It is not yet clear how strategies for acquiring sufficient certifications in the distributed environment compare performance-wise with centralized versions like Advogato [22].

3.13 Semplesh

Semplesh is a system that stores RDF Triples in a number of distributed hash-tables (DHT) e.g. chord. Distributed hash tables conventionally work with document hashes, however Semplesh takes the hash of each possible pair of elements of an RDF triple (Subject, Predicate, Object) and maps it to the remaining element. Thus there are three DHTs each supporting on of the following mappings subj:pred->obj, pred:obj->subj, subj:obj->pred. Thus different queries are directed to the appropriate DHT. In order to perform queries where only one element of the RDF triple is specified, the same DHTs are used but additional wild card insertions are made when triples are inserted. The insert process involves breaking the triple up into three relations, one for each DHT, and then also adding each of these relations three times, so for a triple like "John" "loves" "Mary" being added to the subj:pred->obj DHT would involve three inserts: John:loves->Mary, John:*->Mary, and *:loves->Mary. Thus one can query the DHT for Subject="John" and receive all triples with John as the subject. Naturally the operation of the DHT is such that high popularity triples will lead to a heavy load at certain points in the network. It would also appear difficult to support substring matching.

3.14 HyperCuP

HyperCuP (of Schlosser et al. [30]) uses broadcast search coupled with a multi-dimensional hypercube topology that allows messages to be sent once and once only to each node in the network. Each node in the network has some set of neighbours in each dimension, and messages include information about the dimension along which they have been forwarded. Subsequent transmissions are then restricted to be along higher dimensions, and this ensures that any message will be sent only once to each node. Thus the network diameter will be the $\log_b N$, where N is the number of nodes

in the network, and b is the base of the hypercube. HyperCuP includes a decentralized topology maintenance algorithm that can handle nodes joining and leaving the network and supports routing within partially completed hypercubes. The number of messages generated when nodes leave and join the network is $O(\log_b N)$. An ontology based routing system is proposed whereby a separate hypercube is created and maintained in order to support searches related to a particular concept, and each possible combination of concepts. Thus each node might participate in multiple hypercubes, each of which would provide efficient search for documents/resources associated with that concept, or combination of concepts. Naturally this increases the complexity of joining and leaving the network, and forces nodes to maintain $(b-1)$ times the dimensionality of the hypercube additional connections for each concept or combination of concepts. No simulations of the system appear to have been performed yet, and there is no consideration of malicious nodes.

3.15 YouServ (Formally Usearch)

YouServ, created by Bawa et al. [2], is a distributed search application for personal web servers operating within a shared environment such as a corporate intranet. Almost 1900 people within IBM use the YouServ personal web serving system every week. YouServ enhances web servers with a search component consisting of a content indexer and a query evaluator, that supports keyword-based search.

The indexer regularly monitors shared files, immediately updating its local index when changes are detected. A Bloom filter content summary is created by each peer and pushed to a centralized registrar. When a browser issues a search query at a peer, the peer first queries the summaries at the registrar to obtain a set of peers R in the network that are hosting relevant documents. It then directly contacts the peers in R to obtain relevant URLs. These results are cached for a limited time at the peer that issued the query, which is also responsible for notifying the registrar of the caching activity. This allows other peers that happen to be handling an identical query to locate and return the cached results instead of executing the query from scratch.

The fact that YouServ has been deployed with a significant number of users allows Bawa et al. to perform interesting and useful analyses of their network to assess responsiveness and user satisfaction. YouServ includes other features such as the ability to search within a community context, or over data that has received explicit recommendation by different sub-sets of users. The impact of malicious nodes is minimized given one assumes the co-operative intranet environment in which YouServ is deployed. Bawa et al. also describe how the registrar acts as the limiting factor on the number of nodes that can be supported, indicating that a registrar with a T1-line bandwidth would easily support at least 10000 nodes.

3.16 pSearch

The goal of pSearch [34] is to build a scalable P2P Information Retrieval (IR) system, that has efficiency of DHT systems and accuracy of state-of-the-art IR algorithms. Tang et al. present two algorithms, pLSI and pVSM, which project either a latent semantic index (LSI) or vector space model (VSM, e.g. TFIDF) onto eCAN [37], a hierarchical version of CAN. eCAN improves CAN's logical routing cost to O $(\log(n))$, and takes only routes that closely approximate the underlying Internet topology.

In pVSM, documents, or pointers to documents are stored under hashes of the most heavily weighted VSM terms (keywords). Thus each zone in the CAN network is responsible for a set of keywords, and nearby zones are not necessarily semantically related. Conversely, pLSI stores each document pointer directly against the transformed semantic vector generated by LSI, meaning that semantically related documents will be in close to one another within the CAN network space. A query in pVSM involves contacting all the nodes responsible for the keywords present in the query, while in pLSI the query defines a point in the CAN space, and the query is flooded to all the neighbouring nodes within a small radius.

pLSI simulations suggest that it is able to achieve accuracy comparable to a centralized algorithm while visiting only 0.4–1% nodes in the system. Analysis of pVSM indicates the number of visited nodes is bounded by the number of terms in a query, which is usually small. The authors address various issues such as uneven distribution of document pointers over the nodes, and the global state required to support the VSM and LSI calculations.

4. Discussion

A number of key issues arise when trying to create a distributed meta-data scheme that will provide consistently good results for multiple users. Many of the systems described above attempts to deal with some of these issues, but none of them address them all.

4.1 Representation of the Document

Keyword associations suffice for some applications, but others require more complex representations. How will this meta-data be generated and maintained? The simple solution is to use an automatic approach such as TFIDF, but this limits the representation of the documents to a single standard that might not meet the needs of all users.

For example FASD automatically generates a

TFIDF representation when a document is stored, but TFIDF does not necessarily reflect the desired meta-data of the author, since it is simply a statistical operation. In contrast, Edutella allows documents to be marked up in detailed RDF. Clearly, more complex forms of meta-data make greater storage demands on each individual node in the distributed system, as each node must somehow maintain records of this more complex markup. However more detailed meta-data would seem to be essential if complex queries are to be supported, and a fixed representation of each entity (such as TFIDF) does not allow for differences in opinion about the document meta-data.

4.2 Maintaining up to Date Indices/Routing-Tables

If messages are to be routed according to the contents of their meta-data, how does one maintain the accuracy of the routing tables in every node? What kind of messaging overhead is required to support these updates and how much storage space can be devoted to the routing tables themselves? Given that there are storage space limits a data-removal policy becomes necessary for when these limits are exceeded.

Routing Indices and LimeWire's Query Routing distribute updates as node contents change, and both suggest compression or batch updates to reduce overhead. Conversely, Alpine and NeuroGrid do not distribute routing table information at all; instead they infer routing information from search results. The extent to which routing tables need to be updated will reflect the degree of peer and meta-data churn in the network. In systems that require explicit routing updates, there will be a trade off between the accuracy of the data and the update message overhead. Systems such as Alpine and NeuroGrid that leverage existing messaging to support updates ensure that a highly active system will be frequently updated to take account of changes, but this does not cover situations where there is a significant difference in the frequency of queries and the frequency of change in the underlying meta-data. For example, in some systems the incidence of queries may be low, but the actual meta-data assignments may be frequently changing, in which case updates in response to content change may be more appropriate, such as those employed by YouServ.

In addition there will be practical limits to the storage space available for routing at each node. Clearly the more space devoted the more successful the node is likely to be in acquiring useful data for its users—thus if the quality of the search results is poor one can simply add more storage resources. However the policy employed for deleting data when the storage capacity is exceeded will also have an effect. It would seem profitable for nodes to try and maintain information about high-quality sources of data that are related to the needs of their users. Thus deleting routing information related to infrequently queried meta-data would seem like a good strategy. In a network with many dishonest nodes it might also be profitable to actively maintain information about nodes that were particularly untrustworthy.

4.3 Honesty

Given that indices or routing tables and meta-data are stored in multiple locations how can one be sure that this information is being supplied honestly? What is to stop unscrupulous individuals from marking up their product pitches in meta-data that misleads the user? Or routing queries to inappropriate destinations. Independently of malicious activity there is a fundamental question concerning pure differences of opinion about markup.

Alpine, NeuroGrid and JXTASearch are systems that provide a framework for handling these kinds of issues, although the applications that employ TFIDF could argue that they are attempting to enforce an unbiased approach to markup. However this relies on the assumption that all nodes will actually implement the TFIDF markup scheme, rather than just generating relevance values to meet their own ends. Honesty and differences of opinion about what is the "correct" meta-data for a document become crucial in a distributed network environment. It would seem paramount that nodes should be able to identify which nodes are able to meet their information needs. Whether the underlying motive is dishonest or purely a difference of opinion might be profitably considered, but how to distinguish between the two is problematic. A person trusted to have consistently different opinions from ourselves but who is intrinsically honest will be treated differently from someone who is behaving dishonestly. Assuming that systems can establish a mechanism whereby information providers and consumers can maintain mutually profitable relationships, new concerns emerge. How do new information providers break into these "cliques"? Many of the problems associated with human social networks would appear to be replicated in such systems. Superficial solutions might be to add some degree of probabilistic behaviour to nodes such that occasionally they will take a chance and query nodes with little or no reputation.

Given that each node is maintaining data on the relative trustworthiness of other nodes it would make sense for nodes to restrict access to their resources on this basis. For example a node might be more inclined to forward the queries of another node that consistently supplied useful information. Some systems such as MojoNation (now Mnet) make this relationship explicit by introducing currency that must be handed over in order to make queries, but can be generated by handling the queries of others. It would be interesting to compare the relative merits of a network wide currency, where

doing someone a favour generates credits that can be traded with anyone for anything and those where currency is only good with specific nodes, e.g. a node will only forward messages for other nodes that have done reciprocal favours.

4.4 Integration of New Meta-Data Schemes into Existing Networks

Given that simulations show the potential effectiveness of distributed meta-data schemes, can they be integrated piece-wise into existing peer-to-peer networks?

LimeWire's Query Routing proposal explores the issue of networks where some nodes have implemented a meta-data routing scheme and some have not. Clearly one can avoid this issue by assuming that a network is implemented from scratch. However existing networks have various desirable properties, such as large pools of users that make these considerations important. Out of the systems described above only the LimeWire Query Routing Proposal considers the complexity of trying to add a distributed meta-data search strategy to an existing network, and concludes that further study is required to ensure that any losses will not outweigh the gains.

4.5 Timeliness and Accuracy

Some systems assume/assert the information returned in a query is accurate at the moment a response is generated; others would have no way of knowing and make no guarantees. What are the implicit assumptions made by systems about the lifetime of a query result or document identity?

This is a complex issue, with some systems that use Bloom filters attempting to show that the filters can be updated in time to have query responses reflect the state of the documents in the network. Other systems that avoid explicit updates may easily end up supplying out of date results, i.e. pointers to documents that no longer exist in the network (and potentially out-dated meta-data—document associations). However part of the motivation behind reputation learning, is that reputations will also reflect the availability (as well as the relevance) of the documents recommended in response to particular queries. Although clearly one can provide an accurate recommendation to a document that is not available through no fault of one's own.

Naturally systems that identify documents in terms of hash id do not suffer from this problem in as much as the id—document relationship is constant, still systems like Chord and Freenet have to remove documents when cache limits are reached, but this becomes a document storage issue, rather than one of meta-data.

4.6 Effects of Query Forwarding

Does query forwarding distort the precision/recall/honesty of a result set? And what measures can be put in place to counter ill effects? Does query forwarding effect reach or result set size?

Systems like Alpine avoid any query forwarding by relying on minimal footprint UDP connections to support multiple connections between one peer and the other peers that it intends to communicate with. This allows Alpine to maintain direct statistics on the responses from each peer. Alpine allows peer groups to grow through transitive introductions, so if no one in your peer group can satisfy your query it may take some time before someone can be found who does. NeuroGrid nodes, on the other hand, forward queries that they can't satisfy, potentially expediting the satisfaction of the query. Clearly, forwarding messages can lead to an increase in reach and result set size, but its effect on precision and recall is less clear. The information retrieval metrics of precision and recall are, respectively, the ratio of relevant/non-relevant documents in the result set, and the ratio of relevant documents in the result set and the total number of relevant documents in the system. Query forwarding will plausibly increase recall but its effect on precision is more difficult to discern. At least one system, FASD, ensures that forwarded queries generate only answers that are closer matches than those found so far.

It seems unlikely that query forwarding would particularly affect the honesty of the search results, unless different nodes had different opinions about the honesty of other nodes—a possibility. The only system to use both query forwarding and honesty measures is NeuroGrid, and one can assume that if one is querying high honesty nodes, then subsequent query forwarding will potentially reach nodes considered less honest by the query originator. The solution would appear to be to consider the quality of a node in terms of both the responses it makes to queries, and the performance of the nodes it chooses when it forwards queries.

4.7 Knowing When to Stop Searching

One of the major advantages of a centralized search system is that it can offer guarantees about whether all of the available data has been searched or not. In a distributed search system of potentially unknown size, it is very difficult to be sure that every possible match to your query has been assessed. And given that some query is currently underway it is difficult to ascertain whether a slightly longer wait might generate some more appropriate match to a query. Systems that use DHTs seem to offer the potential to offer some sort of guarantee that if a match exists it will be found. However the use of hashing imposes a particular level

of granularity on the search interface. For example, if we hash keywords and use them in a DHT, it is then difficult to search for documents that have keywords containing a certain substring. One possible solution is then to place hashes of every possible substring from a given keyword into the DHT, but this greatly increases the number of nodes that will contain references to a particular entity, and also increases the amount of time taken to insert a new entity into the system. The need for decentralized search derives from the fact that the amount of data we desire to search over is exceeding the storage limits of centralized systems. It may be that in order to navigate greater volumes of data we must put up with ever increasing degrees of uncertainty about whether we have obtained the best match for our query. That said, it may yet be the case that there is some form of hybrid meta-data DHT scheme that offers the best of both worlds, just waiting to be discovered.

5. Conclusion

It seems clear that the wealth of content made available by peer-to-peer systems like Gnutella and Freenet have spurred many authors into considering how meta-data might be used to support more effective search in a distributed environment. This paper has reviewed a number of these systems and attempted to identify some common themes. At this time the major division between the different approaches is the use of a hash-based routing scheme. Several of the systems described above use Bloom filter hashes to describe the contents of other nodes. The systems that do not use a hash-based approach tend to use more exotic meta-data formats and incorporate some form of reputation measure.

While simulations may show that hash based techniques scale well with increasing network size, they have yet to prove themselves in real peer-to-peer networks. Considerable effort is still required on the simulation level to assess the relative scalability of the different approaches, and there is a particular need for more comparative simulations. Additionally, the anarchic nature of decentralized peer-to-peer networks would seem to demand consideration of how to cope with meta-data reliability and dishonest markup.

References

[1] O. Babaoglu, H. Meling, and A. Montresor, "Anthill: A framework for the development of agent-based peer-to-peer systems," Technical Report, UBLCS-2001-09, Nov. 2001.
[2] M. Bawa, R.J. Bayardo, Jr., S. Rajagopalan, and E.J. Shekita, "Make it fresh, make it quick: Searching a network of personal webservers," WWW2003, Budapest, Hungary, May 2003.
[3] E. Bonsma, "Fully decentralized, scalable look-up in a network of peers using small world networks," Proc. Systemics, Cybernetics and Informatics (SCI), 2002.
[4] C.M. Bowman, P.B. Danzig, D.R. Hardy, U. Manber, and M.F. Schwartz, "The harvest information discovery and access system," Computer Networks and ISDN Systems 28, pp.119–125, 1995.
[5] K. Burton, "Design of the OpenPrivacy distributed reputation system," http://www.peerfear.org/papers/openprivacy-reputation.pdf, 2002.
[6] B. Bloom, "Space/time trade-offs in hash coding with allowable errors," Commun. ACM, vol.13, no.7, pp.422–426, 1970.
[7] J. Chapeweske, "HTTP extensions for a content-addressable Web," http://open-content.net/specs/draft-jchapweske-caw-03.html, Open Content Network Specification, 2002.
[8] I. Clarke, O. Sandberg, B. Wiley, and T.W. Hong, "Freenet: A distributed anonymous information storage and retrieval system," Proc. Workshop on Design Issues in Anonymity and Unobservability, ed. H. Federrath, Berkeley, CA, July 2000.
[9] E. Cohen, A. Fiat, and H. Kaplan, "A case for associative peer to peer overlays," HotNets-I, Princeton, New Jersey, USA, 2002.
[10] A. Crespo and H. Garcia-Molina, "Routing indices for peer-to-peer systems," The 22nd International Conference on Distributed Computing Systems (ICDCS'02), Vienna, Austria, 2002.
[11] F.M. Cuenca-Acuna, C. Peery, R.P. Martin, and T.D. Nguyen, "PlanetP: Using gossiping to build content addressable peer-to-peer information sharing communities," International Workshop on Peer-to-Peer Computing, Pisa, 2002.
[12] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman, "Indexing by latent semantic indexing," J. Am. Soc. Inf. Sci., vol.41, no.6, pp.391–397, Sept. 1990.
[13] P. Deutsch, R. Schoultz, P. Faltstrom, and C. Weider, "Architecture of the WHOIS++ service," http://www.ietf.org/rfc/rfc1835.txt, RFC 1835, 1995.
[14] R. Dornfest and D. Brickley, "Metadata," in Peer to Peer: Harnessing the power of disruptive technologies, ed. A. Oram, pp.191–202, O'Reilly & Associates, 2001.
[15] P. Druschel and A. Rowstron, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," Proc. 18th IFIP/ACM International Conference on Distributed Systems Platforms, 2001.
[16] T. Hoshiai, "P2P communications," HPSR2002 Tutorial, pp.1–62, 2002. http://www.geocities.co.jp/SiliconValley/8143
[17] G. Kan, "Gnutella," in Peer-to-Peer: Harnessing the Benefits of Disruptive Technologies, ed. A. Oram, pp.94–122, O'Reilly & Associates, 2001.
[18] A.Z. Kronfol, "FASD: A fault-tolerant, adaptive scalable, distributed search engine," Princeton University Technical Report, http://www.cs.princeton.edu/~akronfol/fasd/, 2002.
[19] S.R.H. Joseph and T. Kawamura, "Why autonomy makes the agent," in Agent Engineering, pp.7–28, World Scientific Publishing, 2001.
[20] S.R.H. Joseph, "NeuroGrid: Semantically routing queries in peer-to-peer networks," International Workshop on Peer-to-Peer Computing, Pisa, 2002.
[21] B. Krishnamurthy, C. Wills, and Y. Zhang, "On the use and performance of content distribution networks," ACM SIGCOMM Internet Measurement Workshop, 2001.
[22] R. Levien, Attack Resistant Trust Metrics, Draft Doctoral Thesis, University of California, Berkeley, 2002.
[23] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmer, and T. Risch, "Edutella: A P2P

networking infrastructure based on RDF," White Paper, http://edutella.jxta.org/, 2001.
[24] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the web," Technical Report, Stanford University, 1998.
[25] C.G. Plaxton, R. Rajaraman, and A.W. Richa, "Accessing nearby copies of replicated objects in a distributed environment," Proc. ACM SPAA. ACM, June 1997.
[26] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A scalable content-addressable network," Proc. ACM SIGCOMM, 2001.
[27] S.E. Robertson and K. Sparck-Jones, "Relevance weighting of search terms," J. Am. Soc. Inf. Sci., pp.129–146, 1976.
[28] C. Rohrs, "Query routing for the Gnutella network," http://www.limewire.com/developer/query_routing/keyword%20routing.htm, 2002.
[29] G. Salton and C. Yang, "On the specification of term values in automatic indexing," J. Doc., vol.29, pp.351–372, 1973.
[30] M. Schlosser, M. Sintek, S. Decker, and W. Nejdl, "HyperCuP—Hypercubes, ontologies and efficient search on P2P networks," Proc. International Workshop on Agents and Peer-to-Peer Computing (AP2PC), ed. G. Moro & M. Koubarakis, pp.85–96, 2002.
[31] M.A. Sheldon, A. Duda, R. Weiss, and D.K. Gifford, "Discover: A resource discovery system based on content routing," Proc. 3rd International World Wide Web Conference, Elsevier, North Holland Computer Networks and ISDN Systems, 1995.
[32] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," Proc. ACM SIGCOMM'01 Conf., 2001.
[33] A. Sugiura and O. Etzioni, "Query routing for Web search engines: Architecture and experiments," Proc. 9th International World-Wide Web Conference, Foretec Seminars, 2000.
[34] C. Tang, Z. Xu, and M. Mahalingam, "pSearch: Information retrieval in structured overlays," HotNets-I, Princeton, New Jersey, USA, 2002.
[35] D. Watts and S. Strogatz, "Collective dynamics of 'small-world' networks," Nature, vol.393, pp.440–442, 1998.
[36] B. Wiley, "Tristero," http://tristero.sourceforge.net, 2002.
[37] Z. Xu, C. Tang, and Z. Zhang, "Building topology-aware overlays using global soft-state," The 23rd International Conference on Distributed Computing Systems (ICDCS'03), Providence, Rhode, Island, May 2003.
[38] B.Y. Zhao, J.D. Kubiatowicz, and A.D. Joseph, "Tapestry: An infrastructure for fault-resilient wide-area location and routing," Technical Report CSD-01-1141, U.C. Berkeley, 2001.

Sam Joseph is a research associate at the University of Tokyo, where he works on peer to peer and distributed information management systems. Dr. Joseph received a B.Sc. (Hons) in physics with astrophysics at the University of Leicester, UK, followed by a M.Sc. in cognitive science and natural language and a Ph.D. in neural networks from the University of Edinburgh, UK. He is a recipient of the Raymond-Hide prize for Astrophysics and a Toshiba Fellowship. As part of the Toshiba Fellowship he worked on software agents at Toshiba's Research and Development Center in Japan. Dr. Joseph continues to provide consulting services ranging from cognitive science to peer to peer networking, to a number of Japanese technology companies.

Takashige Hoshiai is a senior research scientist supervisor at NTT Network Service Systems Laboratories, in Japan. He holds a Ph.D. degree in Communications and Systems from The University of Electro-Communications, Japan. His research areas are distributed systems, distributed object technologies, real-time systems, agent systems and P2P. Since he proposed a new business model called "Brokerless Model" in 1998, especially, he has studied SIONet architecture that is a solution of P2P platforms.