Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Jan Mischke, Burkhard Stiller

Design Space for Distributed Search $(DS)^2$ - A System Designer's Guide

TIK-Report
Nr. 151, September 2002

Institut für Technische Informatik und Kommunikationsnetze
Computer Engineering and Networks Laboratory

Jan Mischke, Burkhard Stiller:
Design Space for Distributed Search $(DS)^2$ - A System Designer's Guide
Version 1, September 2002
Version 1.1, March 2003; general implications added (Section 7)
TIK-Report Nr. 151

Computer Engineering and Networks Laboratory,
Swiss Federal Institute of Technology (ETH) Zurich

Institut für Technische Informatik und Kommunikationsnetze,
Eidgenössische Technische Hochschule Zürich

Gloriastrasse 35, ETH-Zentrum, CH-8092 Zürich, Switzerland

Design Space for Distributed Search (DS)^2 — A System Designers' Guide —

Jan Mischke[1] and Burkhard Stiller[2,1]

[1] Computer Engineering and Networks Laboratory TIK, Swiss Federal Institute of Technology (ETH Zurich)
Gloriastrasse 35, CH – 8092 Zürich, Switzerland
[2] Information Systems Laboratory IIS, University of Federal Armed Forces Munich
Werner-Heisenberg-Weg 39, D-85577 Neubiberg, Germany

E-Mail: [mischke|stiller]@tik.ee.ethz.ch

Abstract

Important research efforts are conducted in the area of search, lookup, and routing, and are even increasing in light of promises and challenges of peer-to-peer (P2P) systems or the semantic web. To organize these areas of concern, this paper proposes an exhaustive two-dimensional design space that structures and classifies current and facilitates future research. In the functional dimension, it identifies a series of mappings and integrated approaches like keyword lookup or semantic routing. In the structural dimension, design options for each mapping include computational approaches, central or completely replicated tables, classic or symmetric hierarchies, ordered spaces, as well as random structures and topologies. An evaluation of these design options serves as a guideline for system designers and leads to the design of a novel peer-to-peer based keyword routing scheme. It also leads to general implications and recommendations for P2P search design.

Keywords: *Distributed systems, peer-to-peer, design space, keyword search, semantic routing, lookup, distributed tables, overlay topology, symmetric hierarchy*

1 Introduction

A distributed system is a network of nodes and links between these nodes. Particularly in large networks, nodes collectively host abundant amounts of *resources*. Resources include *content* or *information* residing on hosts, like music files, newspapers, or files in distributed file systems, furthermore *hardware resources*, like storage space, computing cycles, network links, or output devices, and finally all kinds of *services*, *e.g.*, specific application software [21].

The difficulty of finding and retrieving or using these resources is increasing with the network size and degree of decentralization. While it was rather easy in times of mainframe computing with only few connected terminals, the move towards completely decentralized peer-to-peer systems with millions of active nodes imposes huge challenges on distributed search and routing. Innumerable efforts have been started to address these issues and design new search systems meeting the requirements of today's peer-to-peer networks. However, a clear structure and delineation of the approaches is yet missing as well as a comparative evaluation. Furthermore, there is no clear statement as to what designs are viable at all and may not even have been looked into.

Consider peer-to-peer file-sharing services as an example. Napster [34] provided a central directory server to enable users to find content - it failed, mostly due to legal issues with its centralized architecture. Hence, Gnutella [6] chose a completely decentralized approach based on flooding - but it can obviously not scale to the millions of nodes expected to join future P2P systems [30]. Chord [1], CAN [26], Tapestry [42], Pastry [9], or AGILE [18] (cf. also [18] for a comparison) designed highly scalable combined lookup and routing systems - however, their highly structured approach makes them vulnerable to malicious users and makes keyword search a non-trivial task. In the world wide web, search engines like Google, together with DNS and IP routing [25] jointly support information search and retrieval and proved to be highly successful - but they rely on hierarchies and central infrastructure or authorities that are not available in decentralized P2P environments. And even despite their success in the web, alternative or supplementary approaches like TRIAD are being developed and proposed [10].

So several questions arise: What is the best design to choose for a specific application? What are the fundamental parallels or differences between these and further approaches that make them better or less well suited for one or the other application? To what extent can application-specific requirements be met, and what is the effect of choosing one or another design, what are the trade-offs? How to effectively approach new system design in a highly structured way? And, finally, are there any fundamentally new approaches yet to be discovered and developed that may achieve significant performance leaps?

This paper addresses these and further issues. It gives clear guidance to system designers on which design to choose or what kind of system to develop for their specific needs. And it helps researchers to classify existing systems and current efforts, to identify blind spots in the design space, and to structure future research.

The following design space first yields a clear separation of two fundamental design dimensions, the functional and the structural dimension. Regarding the functional dimension, Section 2 gives an overview of the necessary steps required to find and use or retrieve resources in a distributed system and identifies possible short-cuts. As to the structural design dimension, Section 3 spans the possible solution space in a top-down approach. Section 4 discusses methodologies as well as advantages and disadvantages of the design options

and extensively tests the framework by organizing current solutions and efforts accordingly, thus also presenting related work. A more detailed evaluation of functional and structural design options along a set of major requirements follows in Section 5. This evaluation forms the basis for a guide to search system design in Section 6. The same Section also tests the guide along a peer-to-peer application, leading to the proposition of a novel keyword routing scheme. Section 7 derives implications for future directions in research on peer-to-peer search algorithms before Section 8 concludes.

2 The Search Process and Functional Design Space

For a thorough understanding of the functionality required to find and use or retrieve resources and for possible optimizations, it is required to disaggregate the search process and identify necessary steps and possible short-cuts. Figure 1 shows the process from *keywords* over *names* and *addresses* to the *path to target node* hosting the desired resources.

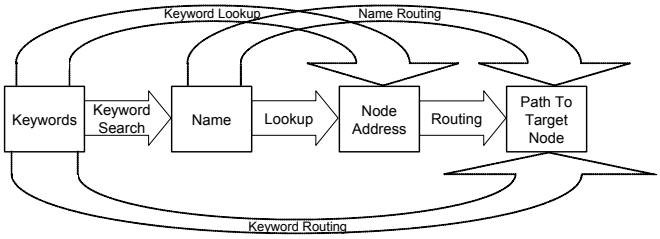

Figure 1: Search in Peer-to-peer or other Distributed Systems

In most cases, a user will want to specify what he or she is looking for in terms of *keywords*. In the simplest case, keywords are just one or more terms appearing in the desired content or describing the desired resource. More sophisticated approaches apply content/resource meta information based on attribute-value pairs. The Resource Description Framework, RDF [28], attempts to standardize this meta information. *Keyword search* describes the functionality of mapping the resource meta information onto one, or, in the case of multiple matching resources, several unique *names* or identifiers in the network. The most classical example of such names are the Uniform Resource Locator, URL, or file names in a Unix file system. *Lookup* maps unique names onto *addresses* in the network. Addresses specify the network location of the node hosting the resource with a given name. This can be the IP address of the host or, *e.g.*, an address in an overlay network as it is common in peer-to-peer networks. Finally, *routing* is the process of finding a *path* and moving queries to the *target node*. Routing can also take place on a physical or an overlay network.

Three short-cut mechanisms can help optimize search. *Name routing* is becoming more and more common in P2P systems and combines the (distributed) lookup of the target node address with path identification and query forwarding to that node. *Keyword lookup* returns one or more addresses of nodes hosting resources with given keyword descriptions. Napster is the most prominent example. Finally, *keyword routing* directly routes towards a node hosting specified resources. Keyword routing is sometimes also called *semantic routing* or *content routing*.

3 The Structural Design Space for Existing and Future Systems

With the search process defined and disaggregated, it becomes obvious that searching requires a series of mappings, from the keyword space to the name space to the address space to the space of paths to nodes. In order to assist system design, the options for such a mapping in distributed systems are depicted in Figure 2.

A mapping is defined through a *computation* or a *table*. Computation is difficult to achieve, but some attempts have been made, usually involving hashing. More widely adopted are *tables* with entries for the desired search items, *e.g.*, a node address for each valid name. Mapping then comes down to finding the desired table entry and looking up the associated value. In a distributed environment, a table can either reside on a *central* entity like a search engine server, or be completely *replicated* on each node, or be *distributed* among the nodes.

Distributed tables are probably most interesting and challenging in that they require for each mapping to collaboratively find and contact the node that has the desired information or table entry. Two important aspects distinguish distributed table approaches: the *structure of the table*, i.e. the distribution of table entries to nodes, and the physical or overlay *topology* of the network. The distribution of table entries either happens at random or in a well-designed process leading to a clear target table structure; the same applies for the distribution of links and, hence, the topology. Whether the table structure and topology are designed and aligned, or both random, or at least one of them designed but not aligned with the other, has substantial implications on search.

In a *random table structure and random topology,* it is natural that each node at least carries information about itself, i.e. its address, the names of its resources and content, and corresponding keyword descriptions. In addition to information on their own tables, nodes may have knowledge on the table entries of their *neighbors*, i.e. the nodes they directly know about and may contact for search, in an aggregated or non-aggregated form. The knowledge on neighboring table entries will in some cases be restricted to the direct neighbors, but can also involve *recursion*: An arbitrary node A not only learns about the table entries of its neighbors B_i, but also through B_i about B_i's neighbors C_{ij}, C_{ij}'s neighbors D_{ijk}, and so on. This way, nodes eventually know about all keywords, names, or addresses in the direction of each neighbor in a usually aggregated way.

Rather than keeping explicit knowledge on neighboring table entries, nodes can exploit *implicit knowledge* when the table distribution and topology follows a clear and *aligned structure* that every node knows. The most common approach is certainly the *classical hierarchy.* A root node informs about table areas represented by a number of second-level nodes. The second-level nodes, in turn, delegate to third-level nodes for sub-areas within their own area, and so

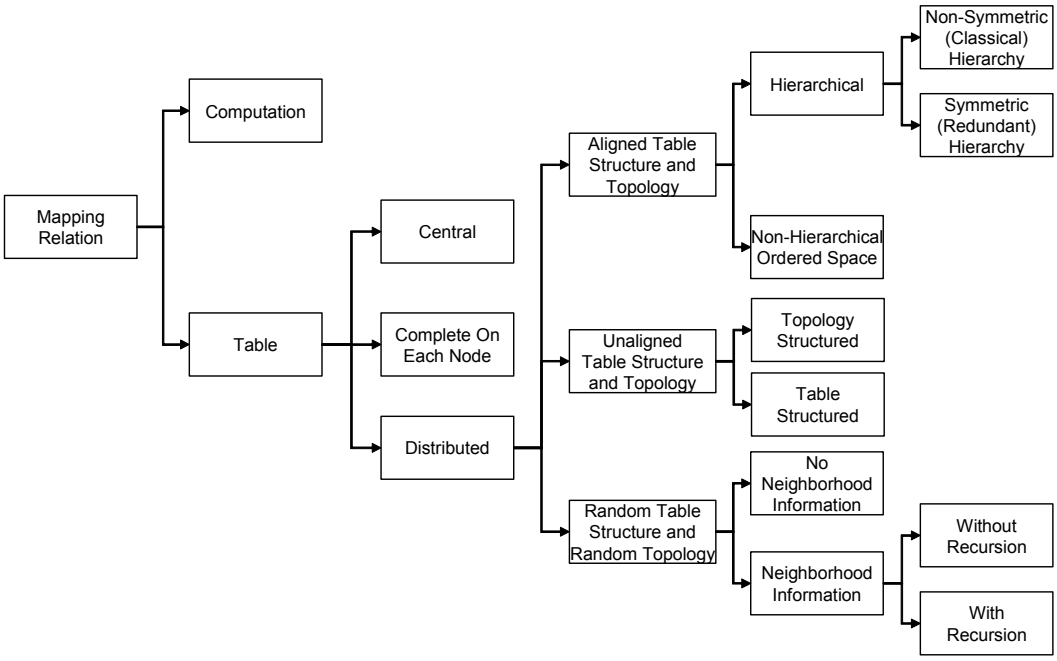

Figure 2: Design Space for Mapping Relations in Distributed Systems

on, until a request finally reaches the leaf node responsible for the desired entry. Particularly in the quest for scalable peer-to-peer search algorithms, "*symmetric hierarchies"* have been created by adding *redundancy.* In symmetric hierarchies, every node can become the root or be on any other level of the hierarchy. This can be achieved by replicating the root information on table areas on each node as well as the second-level information on sub-areas etc. Note that on each node, only the sub-area information within one top-level area is available rather than that within all top-level areas; otherwise the table on each node would be complete. Also note that nodes acting as a root will usually point to different neighbors for the second level table areas (and so on for all remaining levels) as there are multiple different options due to the replication. The symmetric hierarchies being created show structural similarities with k-ary n-cubes, see [18] for a more detailed discussion. *Non-hierarchical structures* are also possible and available. In an *ordered space*, the table is split into consecutive areas. Each of the areas is represented on one node. The nodes, in turn, are ordered in the same way, i.e. neighboring table areas reside on neighboring nodes. Examples of such spaces are rings or Euclidean spaces.

Unaligned table structures and topologies occur when either the topology is designed but the table structure random, or the table is distributed according to a clear structure but the topology is random, or both table and topology are clearly structured but in different ways. While the first case can be advantageous for performance improvements compared to a completely random approach, the second case is particularly used to allow aggregation of table area information. It appears difficult to gain from the third case.

4 Functional and Structural Design Space: Methodologies and Sample Systems

This section presents a framework for design options in distributed search and gives an overview of existing systems. By briefly discussing key requirements, advantages, and drawbacks, it explains the rationale for choosing a specific design. Furthermore, blind spots in the design space will be identified where further research may lead to entirely new systems with significantly improved performance.

Table 1 classifies existing approaches into the design space categories. While it represents the best of the authors' knowledge, it primarily serves as the basis for detailed discussions, a proof of concept, and the outline of major research dimensions.

4.1 Computational Approaches

Computational mapping is very efficient in that it involves neither large tables to reside in memory nor bandwidth-consuming query messages to be sent. However, it is difficult to achieve as it requires that all possible outcomes of the computation be allowed in the target range, i.e. name space, address space, or space of routes. Constantly changing target spaces or value ranges, due to the addition or removal of nodes and node addresses, or resources with their corresponding names, limit the applicability of computational approaches. Some have been made, however, usually involving hashing, and circumventing the problem by simply defining name or address spaces such as to cover all possible computation outcomes. This is impossible, however, for routing in dynamic environments, as the paths to nodes have to exist and cannot simply be defined.

INS/Twine [2] builds attribute-value trees from complex resource descriptions and disaggregates them into strands of variable length. Hashing is applied to map the strands onto 128-bit names (which are not necessarily unique in INS/Twine). Chord's name routing algorithm ([1], see below) completes the search.

4.2 Centralized Tables

Central tables are very bandwidth-efficient and incur little overhead. However, they require that a central entity have

Table 1: Search Systems in the Design Space

Design					Keyword Search	Lookup	Routing	Keyword Lookup	Name Routing	Keyword Routing
Computational					INS/Twine[a]	n/a	n/a	n/a	n/a	n/a
Table	*Central*				Search engines, web directories	Phone directories[a]	Star topologies	Napster	Load balancing hub	n/a
	Complete on Each Node				Groove[a]	NIC's Hosts.txt	(Overlay) complete mesh	n/a	n/a	n/a
	Distributed Table	*Aligned Table Structure and Topology*	*Hierarchical*	*Classical*	n/a	DNS	IP	n/a	n/a	TerraDir
				Symmetric (Redundant)	n/a	n/a	n/a	n/a	Pastry, Tapestry, AGILE	SHARK
			Non-Hierarchical Ordered Space		n/a	n/a	n/a	n/a	CAN, Chord	n/a
		Unaligned Table Structure and Topology		*Topology Structured*	n/a	n/a	n/a	n/a	HyperCuP[a], Brocade[a], Morpheus, Kazaa, LimeWire[b], BearShare, Clip2 Reflector	LimeWire[b]
				Table Structured	n/a	n/a	BGP, RIP	n/a	TRIAD/NBRP	n/a
		Random Table Structure and Random Topology	*No Neighborhood Information*		n/a	n/a	Reactive MANET, DSR, AODV	n/a	Random Walk, Gnutella, Expanding Ring	Random Walk, LimeWire[b]
			Neighborhood Information	*Without Recursion*	Manual http-Browsing	n/a	ZRP[a]	P2P relational database	Freenet	Best match
				With Recursion	n/a	n/a	Bellman-Ford	n/a	Variants of Bloom filters	Semandex Netlink, Bloom filters, e.g. LimeWire[b]

a. Only partial fit into category; see text for explanation.

b. LimeWire proposes multiple add-ons to Gnutella and is listed subsequently multiple times in the table.

trust, reliability, and authoritative information access necessary to own the central table. Furthermore, a possible outage of a central server represents a considerable risk for the entire network.

Web search engines like Google apply inverse indices to provide URL names based on keywords. In the non-IP world, phone directories return the phone number, i.e., address, when provided with a name. In star topologies, the central server has the routing information of the entire network and is able to route and forward packets when provided with an address. Napster operates central servers to identify addresses of peers where content files with a file name containing given keywords are stored [34]. Many load balancing hubs route towards a specific server in a server farm based on the URL (name) of the request.

4.3 Completely Replicated Tables

Key advantages of *complete replication* of tables on each node are the increased autonomy and fault tolerance in the system when compared to central tables while keeping the simplicity and bandwidth-efficiency. However, replication and synchronization issues as well as high memory needs usually restrict the approach to small tables and networks.

Collaboration tools like Groove synchronize keyword, name, and address information as well as actual objects or object updates on all nodes [12]. Subsequently, all information necessary for, *e.g.*, keyword search, is available on all nodes, even though the system is based less on a reactive search but more on proactive synchronization. As a predecessor to the Domain Name System (DNS), the Network Information Center (NIC) distributed a file, hosts.txt, to all internet hosts for translation of domain names into IP addresses [25]. Complete information on routing corresponds to a complete mesh, either physical or as a logical overlay, being created.

4.4 Distributed Structured Tables with Aligned Topologies

Classical hierarchies are very efficient for searching and, in contrast to central tables, allow for delegation of responsibility. However, they require an equally hierarchical topology and source domain, i.e. keyword space, name space, or address space, in order to work efficiently.

DNS applies hierarchically organized domain names and an equivalent hierarchy of domain name servers to yield an IP address when asked about a domain name. The hierarchy of IP addresses is exploited for routing in the Internet. Finally, TerraDir [38] organizes all content in a hierarchical keyword structure. For each content item or keyword, a vir-

tual node is created, enabling keyword routing towards that node along the hierarchy.

Symmetric (redundant) hierarchies combine the advantages of a classical hierarchy with the symmetry and fault tolerance requirements of a peer-to-peer system, at the cost of additional redundancy in the system and complex node and resource insertion and removal.

In Pastry [9] and Tapestry [42], content names and IP addresses of nodes are hashed onto the same numerical identifier (ID) space, such as to combine name space and address space in the overlay network; this allows name routing when making that node responsible for holding a resource or a link to it that is closest to the resource in the ID space. The hierarchy is created through a digit representation of the ID to a base value and an association of each digit with one hierarchy level, starting from the last (Tapestry) or the first digit (Pastry), respectively. AGILE (Adaptive, Group-of-Interest-based Lookup Engine, [18]) follows a similar approach as Tapestry but introduces an additional three-level hierarchy for the resource description. Even though motivated through performance improvements (pruning), this is already a step towards symmetric hierarchy-based keyword routing. This approach will be extended in SHARK (Symmetric Hierarchy Adaption for Routing of Keywords, cf. Section 6.2).

Also popular for name routing in peer-to-peer systems is the *non-hierarchical ordered space* approach. The prerequisite here is that source domain, i.e. keywords, names, or addresses, and nodes can be arranged in the same totally ordered, non-hierarchical space.

Chord [1] hashes resource names and node IP addresses to a 128-bit ID. The IDs are arranged in a circle with the predecessor node of a resource ID being responsible for providing the resource or a link to it. Fingers are used as short-cuts to prevent the name routing mechanism from moving around the circle in unit steps. In CAN (Content Addressable Network, [26]), hashing is similarly applied to map resource names onto an ID in a d-dimensional torus. Nodes distribute responsibility for the ID space among themselves and maintain virtual links to all direct neighbors in the torus. Queries for a name, i.e. ID, can then at each node easily be routed into the optimum direction. To the author's knowledge, there is no proposal yet to apply an ordered space to keyword routing, except for extensions of the HyperCuP approach (cf. below, [33]), even though it appears promising particularly for peer-to-peer networks.

4.5 Unaligned Distributed Table Structures and Topologies

Search based on *unaligned table structures and topologies* is most common where a structuring of the table appears prohibitive yet a clear topology improves system performance. This particularly applies to widely-spread peer-to-peer systems, where free-riding and non-trustworthiness of some peers inhibit distributing responsibility for some table entries beyond the corresponding resources' owner.

HyperCuP builds a hypercube topology of nodes to support efficient flooding [33]. An extension arranging resources in the same hypercube space as the nodes which is also proposed by the authors would move it to ordered-space keyword routing. Hierarchies with "landmarks", "supernodes", or "ultranodes" are introduced into many peer-to-peer name routing systems to reduce latency, like in Brocade [43], or to improve scalability, like in Morpheus/Kazaa, LimeWire, BearShare or Clip2 Reflector [41]. In these systems, the landmarks or supernodes replicate address and name information for all subordinate nodes and act as gateways or proxies for name requests between the subordinate nodes and the remaining network. However, as only a hierarchical overlay topology is applied, but not an equivalently hierarchical address or name space, the benefit of the hierarchy is limited.

The Routing Information Protocol (RIP [14]), Border Gateway Protocol (BGP, [27]), and TRIAD/NBRP (Translating Relaying Internet Architecture integrating Active Directories/Name-Based Routing Protocol, [11]), in contrast, build on *hierarchical address and name spaces*, respectively, in order to aggregate information, while the *topology* can be *random*. RIP is a distance vector protocol where each router advertises the number of hops to other routers in the network. The Border Gateway Protocol adapts the approach by having routers only advertise reachability without distance information. Address prefixes assist aggregation wherever possible and constitute the hierarchical element in BGP and RIP. The TRIAD/NBRP system essentially extends the BGP approach to name routing, aggregating through name suffixes [11].

4.6 Random Distributed Tables and Random Topologies

Even though *random table structures* for mapping relationships in search appear less sophisticated than aligned structures and do not allow to exploit implicit structural knowledge, there are a couple of advantages to this approach. The maintenance burden for creating and keeping an explicit table structure can be too high, particularly in fast-changing environments like mobile ad-hoc networks. This also leads to issues regarding fault-tolerance: if the structure is not correct in algorithms that rely on it, queries may not be successful. Finally, structured approaches require a high degree of collaboration and trust. Unless ownership for certain resources referred to in table entries coincides with the assigned responsibility for these table entries, resource owners and search requestors have to rely on third parties to provide correct information. Even though this coincidence applies for DNS, in many cases, like Pastry or Tapestry, this is not the case.

The most simple form of a randomly distributed table lets each node only maintain a table of keywords, names, and addresses (if several on a node) of its *own resources without neighborhood information*. This approach is extremely simple and helpful in environments changing so fast that knowledge about neighbors usually becomes stale before it is used. However, for all mappings, it requires to either arbitrarily choose neighbors to send requests to, or, more commonly, flood the entire network.

Reactive routing mechanisms in mobile ad-hoc networks (cf. [35]), like Dynamic Source Routing (DSR, [15]) or Ad-hoc on-Demand Distance Vector Routing (AoDV, [24]) apply flooding. The approach is also used in the Gnutella

name routing [6] and its extension to keyword routing proposed within LimeWire [39], [40]. In expanding ring searches, the requestor is contacted before each additional request forwarding to check whether the desired object has already been found such as to allow early termination of the query flooding. Multiple *random walks* with termination checking can drastically reduce the number of messages due to the finer granularity of node visits and reduced duplication of messages, and, hence, improve bandwidth scalability [16]. The improvements, however, require sufficient replication of objects and come at the cost of significantly increased latency.

Direct neighborhood information on each node can improve query forwarding decisions within a distributed table. However, unless flooding is used, the approach remains indeterministic as to whether a result can be found in the direction of a neighbor.

Hyperlinks in http provide users with names, i.e. URLs, of resources on neighboring nodes; they can be used for manual keyword search or browsing. The Zone Routing Protocol (ZRP, [13]) for routing in mobile ad-hoc networks stores path information to neighbors within a surrounding zone but applies flooding beyond that zone. Of course, only one node per zone needs to be contacted for the flooding. Peer-to-peer based relational databases as proposed in [3] and return addresses of or links to neighboring databases when searched locally. In Freenet [22], name routing is based on hashes of file names or content information. Each node forwards a query to the neighbor storing content with a hash ID numerically closest to the request. The approach converges due to Freenet's aggressive caching strategy.

Recursive neighborhood information, usually in a very aggregated form, makes a random table structure search deterministic while avoiding flooding. However, the synchronization overhead incurred can be substantial. Specific attention is due in networks containing loops in order to avoid a count-to-infinity problem.

In distance-vector or Bellman-Ford IP-routing [4], [10], routers advertise their distance to other routers. Even though in the stable state, every router has complete information on distances and next hops towards a peer router, information is aggregated (and distributed) in that only the next hop but not the entire path is stored. For name and keyword routing in peer-to-peer networks, various variants of Bloom filters (cf. [5]) have been proposed to aggregate and compress information on resources in the direction of each neighbor. Put simple, one bit is set in a word for each name occurring in a certain direction. Rhea and Kubiatowitcz suggest attenuated Bloom filters storing name information up to d-levels of depth with weights decreasing with distance [29]. For keyword routing, Prinkey proposes standard Bloom filters in tree topologies with aggregated signatures of a branch, i.e. the Bloom filter bits represent the hashed keywords present in a tree branch. LimeWire modifies the proposal to cope with arbitrary topologies [31] by adding the number of hops to a resource when propagating the keyword routing information. Crespo and Garcia-Molina [7] suggest to store and propagate the number of matching documents for each keyword, either together with the number of hops to a document (hop count routing indices), or weighting the number of documents with a cost function depending on the distance (exponentially weighted routing indices). Semandex Netlink routers exchange aggregated XML-based user profile information to subsequently be able to route information to interested users based on meta-information [36], [37], [23].

4.7 Hybrid Approaches

In addition to pure approaches, *hybrid approaches* are possible and used, like Brocade [43], where a central table aggregates information within a group on a landmark router, while a symmetric redundant hierarchy is proposed between the landmark routers.

5 Evaluation of the Design Options

The design space presented in the previous Section leaves the system developer with multiple alternatives. It is impossible to make a general statement as to which designs are best. Moreover, different requirements and importance of requirements characterize various systems. Hence, this section focuses on discussing in detail advantages and drawbacks along a set of major requirements, lying the foundation for structured and conscious future design decisions.

5.1 Functional Design Space

The functional design space can be condensed into two major choices: (a) build a disaggregated search involving the separate steps keyword search, lookup, and routing, or (b) take one of the integrated approaches of keyword lookup, name routing, or keyword routing. The two options will be compared by identifying the advantages of each of them with respect to the other one.

Integrated Approaches

The key advantage of integrated approaches, their *efficiency*, is already reflected in their current deployment mostly for peer-to-peer systems: only one or two mappings are required rather than three. Particularly in widely distributed tables (rather than computational, centralized, or strictly hierarchical approaches), each mapping requires the collaboration of many nodes and incurs high *bandwidth demands* and *latency*. Integrated approaches avoid a duplication of highly similar mapping functionalities.

Keyword or name routing also vastly simplify the integration of sophisticated mechanisms as they allow *keyword-* or *name-based re-routing* and, hence, make the system transparent to path changes, *e.g.,* due to broken links, address changes, *e.g.,* due to site outage or caching and replication, and, in the former case, even name changes, *e.g.*, due to addition or removal of content for a given keyword. This can also alleviate *real-time search*.

Disaggregated Approaches

A decoupling of keyword search, lookup, and routing shows the set of following advantages:

- *Reusability*: Each mapping can be used separately to support a wide range of services; *e.g.*, IP routing can not only be used for keyword search but also to create virtual private networks (VPNs) or enable multimedia multicast.

- *Innovation*: Innovation in one area does not affect another area; *e.g.,* improvements to Google's search algorithms are independent of any router upgrades.
- *Simplicity*: Devices for each step can be simpler and more specialized than for integrated systems, *e.g.*, ultra-fast hardware-based IP routers. Similarly, software engineering is alleviated, also increasing the *maintainability* of the system.
- *Horizontal and Vertical Variety*: Different choices for keyword search, lookup, and routing smoothly interoperate, both horizontally at each step, *e.g.*, central Google-type keyword search in parallel to peer-to-peer based search like Infrasearch, as well as vertically across the three steps, *e.g.*, central search and hierarchical lookup and routing.
- *Ownership Separation*: Each mapping can be offered by a separate entity, allowing more competition leading to higher efficiency and innovation, potentially also hampering censorship.
- *Delegation*: Almost a consequence of ownership separation and vertical variety, but extremely important, is the possibility to logically separate name space and address space, *e.g.*, independent allocation of and even delegation of responsibility for IP address space and domain names.

5.2 Structural Design Space

Table 2 presents an overview on system design requirements and the degree to which they are met by various structural design options. Note that many system developers have added more details to their design-specific features that address shortcomings or fortify strengths of the system. Table 2 can thus only be regarded as a rough guideline. Major advantages and prerequisites of each system have already been highlighted in Section 4. An explanation and definition of these requirements is given below, extending prior work in [17].

- *Manageability and Control*: How hard is it to control and manage the system, i.e. how complex is it, how much maintenance does it require, and what level of control can be exercised?
- *Coherence*: Does the system deterministically find authoritative information, or does it behave in an indeterministic way, or is it prone to retrieve stale replicated information?
- *Extensibility*: How easy or difficult is it to add resources or nodes to the system?
- *Fault Tolerance and Adaptability*: How severely is the system affected by a fault and how easily and quickly can it cope with system changes like node joins or leaves?
- *Scalability*: To what limits can the system grow at reasonable performance, particularly with regards to bandwidth and latency, but also memory and processing load on average nodes as well as hot spots?
- *Publish Autonomy and Security*: Is information mostly kept at the resource owner, can even responsibility for the corresponding name or address space be delegated?
- *Search Autonomy and Security*: Does searching require the collaboration of many, particularly untrusted parties?
- *Infrastructure Independence*: How independent is the system from shared infrastructure like central servers? This property is particularly required for peer-to-peer networks; here, the system cannot rely on shared infrastructure other than basic Internet services as it is usually not available.
- *Special Prerequisites*: As already discussed in Section 3, some designs have special requirements. A computational approach requires that all computation results lead to valid values in the target range of the mapping, *i.e.* name space, address space, or space of paths. Hierarchical approaches require source domains of the mapping, *i.e.* keyword space, name space, or address space, to be hierarchical. Finally, a non-hierarchical ordered-space approach requires an arbitrary order other than a hierarchy to be imposed on the source domain, usually a linear order.

6 Guideline and Sample Application

This section will draw conclusions from the design space and its evaluation by providing system designers with a brief application guide (Section 6.1) and by illustrating the approach along a sample peer-to-peer system and creating a novel keyword routing approach (Section 6.2).

6.1 A 5-step System Designers' Guide

This Section outlines a five-step high-level procedure on how to methodologically apply the design space to derive an optimum search algorithm for a specific application. Each of the steps obviously requires detailed examinations of requirements and algorithms.

1. *Define Requirements*: Define all requirements of your system in detail, particularly based on those requirements evaluated in Section 5. Categorize them into obligatory and optional ones and further prioritize within the categories.

2. *Choose Functional Design*: Based on Section 5.1, trade off an efficient integrated approach with dynamic re-routing vs. a disaggregated approach with the indicated advantages of decoupling.

3. *Establish a Shortlist of Structural Design Options*: Consult Table 2 to determine the best-suited design choices for your proprietary requirement ranking.

4. *Reconsider Functional Design*: If your shortlist includes designs with special prerequisites toward the source domain or target range that you cannot meet, reconsider your initial choice of functional design and find out if you can meet them with a different level of disaggregation.

5. *Choose or Craft Search System*: Consult Table 1 and further literature to identify candidate systems. Evaluate them in detail, including their specific features, and compare to your requirements. Beware that the more detailed evaluation you perform might lead you to reconsider your choice of functional or structural design. Choose an existing solution, craft one from multiple existing solutions, or develop an entirely new one.

Table 2: Evaluation of Structural Design Options[a]

Design					Manageability and Control	Coherence	Extensibility	Fault Tolerance & Adaptability	Scalability	Publish Autonomy and Security	Search Autonomy and Security	Infrastructure Independence	Special Prerequisites[b]
Computational					++	++	--	--	++	++	++	++	yes
Table	*Central*				++	++	-	-	- or +[c]	- or ++[d]	- or ++[d]	--	-
	Complete on Each Node				--	--	0	++	--	++	++	++	-
	Distributed Table	*Aligned T.S & Topology:*[e]	*Hierar-chical*	*Classical*	+/0	+/0	++/+	-	0 or ++[c]	++/- or ++/++[d]	-/- or ++/++[d]	-	yes
				Symmetric	0/-	0/-	+/0	+	++	++/--	-	++	yes
			Non-Hierarchical Ordered Space		0/-	0/-	+/0	+[f]	++[f]	++/--	-	++	yes
		Unaligned T.S. & Topology			tbd.[g]	tbd.[g]	tbd.[g]	tbd.[g]	tbd.[g]	tbd.[g]	tbd.[g]	tbd.[g]	tbd.[g]
		Random T.S. & Random Topology	*No Neighborhood Information*		--	++	++	++	--	++	-	++	-
			Neighbor. Info.	*Without Recursion*	-	+	++	+	-	++	-	++	-
				With Recursion	-	-	+	+	+ or -[h]	0	0	++	-

a. -- very weak/low ... ++ very good/high
b. See text for explanation
c. If additional servers allowed and possible
d. If central entities regarded as trusted and collaborative
e. x/y: First value, if table structure and topology reflect ownership structure, second value if separate from ownership structure
f. In existing systems CAN and Chord, depends on system details
g. To be defined, depends on system details.
h. Depending on level of information aggregation

Upon completion of this procedure, a design should have been selected that is optimum with respect to the individual requirements of the application, in a design space believed to be exhaustive.

6.2 Implications for a Novel Keyword Routing

The validity and usefulness of the design space and the 5-step guide are very briefly demonstrated along their application to a concrete problem and the conception of a novel system. Within the MMAPPS project ("Market Management of Peer-to-Peer Services", [20]), inter alia a search system is to be designed as described below.

1. *Requirements*: The system absolutely has to scale to and be efficient up to several millions of nodes and must not depend on shared infrastructure other than basic Internet connectivity, as there is no entity interested in supplying such an infrastructure. Furthermore, it has to be easily extensible and highly adaptable and fault-tolerant as resources are expected to frequently be added and removed (cf. [32]). Extensive caching of popular content make dynamic re-routing facilities highly desirable. The system is supposed to run on a service-specific overlay network. Therefore, reusability, innovation, horizontal variety, and ownership separation appear less important than this may be the case in other systems. Finally, as peer nodes usually are complex multi-purpose personal computers, device simplicity and specialization seem to be of little importance.

2. *Functional Design*: Overall, the requirements stated above and the functional design space strongly recommend integrated keyword routing.

3. *Shortlist of Structural Design*: The evaluation in Table 2 with respect to the prime requirements, scalability and infrastructure independence, recommends three options, a symmetric redundant hierarchy, a non-hierarchical ordered space, or a random structure with recursive neighborhood information. For this specific case, the advantage in terms of scalability of the first two designs outweighs the advantage of the latter one in terms of autonomy, particularly since additional peer incentives and the creation of a hierarchy that reflects the ownership structure are contemplated to improve on this dimension.

4. *Reconsider Functional Design*: Assuming that a hierarchical or ordered keyword space can be built, a change in functional design is not necessary.

5. *Craft System*: Table 1 indicates that a keyword routing based on a symmetric redundant hierarchy or a non-hierarchical ordered space do not yet exist. This blind spot will be filled by SHARK, Symmetric Hierarchy Adaption for Routing based on Keywords. This approach will extend AGILE's symmetric hierarchy for name routing and build on a hierarchical keyword ontology to allow efficient and highly scalable keyword routing. Virtual nodes based on interest profiles will increase the efficiency and the autonomy of nodes in the system.

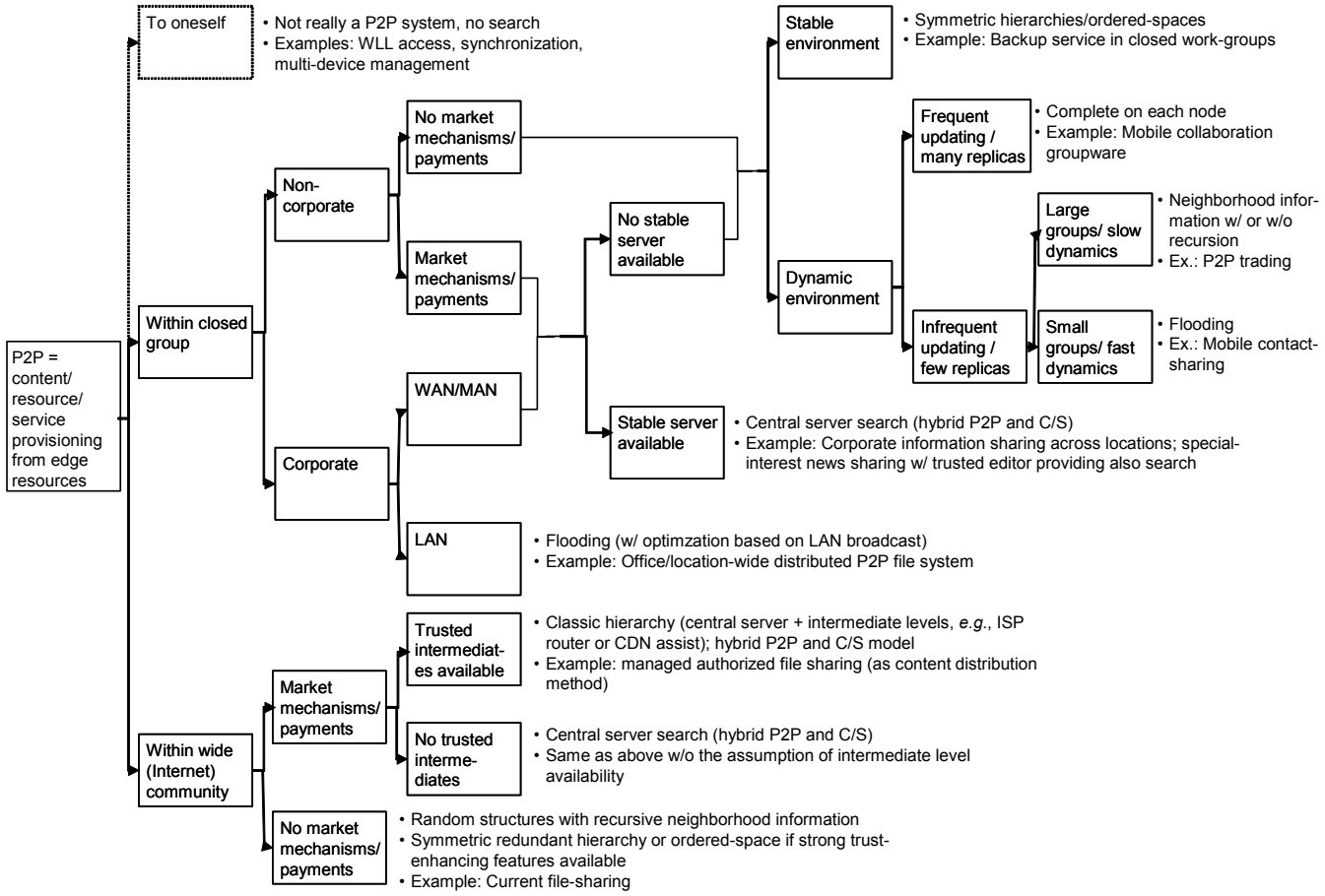

Figure 3: P2P Deployment Scenarios and Recommended Search Systems

7 Search in Peer-to-peer Networks - General Implications and Recommendations

Apart from the implications of the design space on specific systems, a few general trends in search can be observed and discussed, focusing on peer-to-peer networks that have gained much attention due to their explosive proliferation. This sections starts with trends and implications in the functional dimension before exploring the structural dimension.

7.1 Implications on the Functional Dimension

Symmetric hierarchy- or ordered space-based name routing approaches or *distributed hash tables* (DHTs) have recently been a hot research topic for peer-to-peer systems, leading to systems like Pastry [9], Tapestry [42], AGILE [18], Chord [1], or CAN [26]. In absence of central keyword searching facilities as provided in the world-wide web, however, name routing seems mostly helpful for backup or storage systems like the Cooperative File System (CFS, [8]) where the requestor exactly knows the name of the file to be retrieved. Integrated keyword routing, in contrast, would exhibit the following benefits. On the one hand, efficiency is key; bandwidth demand and latency can, depending on the system, roughly be cut in half when integrating distributed keyword search and name routing/lookup. On the other hand, names seem to be of much less importance in P2P than in the web, so that there is no obvious reason to promote them as an in-between step rather than directly retrieving the addresses corresponding to the desired keyword. Whereas names in the web reflect the ownership structure and hierarchy of administrative domains and assist manual browsing, there is no such structure or browsing functionality in P2P, at least not in current systems. Furthermore, the introduction of logical names (URLs) in the web serves to decouple keywords from physical network addresses, such that sometimes fast changes in the address structure due to reconfigurations or server failures can be handled by a close-to-realtime lookup system (DNS) rather than by search engines with an actualization time on the order of days or weeks. For P2P systems, however, the fast changing nature of the system with respect to both, the overlay topology and participating hosts as well as the content or services available, make dynamic re-routing directly based on keywords highly desirable.

7.2 Implications on the Structural Dimension

While the symmetric hierarchy- or ordered space-based approaches mentioned above are highly scalable, they cannot meet some typical peer-to-peer requirements like arbitrary content search in largely untrusted file- or resource-sharing services due to security issues. Random structures with recursive neighborhood information are better suited for these environments.

To generalize these observations and yield recommendations for arbitrary P2P systems, Figure 3 shows a breakdown of possible P2P deployment scenarios as they are relevant for choosing a search or lookup design.

P2P systems are defined as hardware resource (processing, storage, bandwidth), content, or service provisioning among nodes at the edge of the Internet. Whether the system is restricted to closed groups with some form of access control or open to a wide Internet community (like typical file-sharing systems), has strong implications on the optimum search methodology.

In the latter case, trust among peers is difficult to achieve, and scalability usually of high importance. Web search engines and similar systems prove that central systems can meet user demands, and a centralized approach leads to little bandwidth overhead and low latencies. Hierarchies are even more scalable (and lower latency) but they require that nodes be available for the intermediate levels of the hierarchy between central search server and peers, e.g., through router or gateway upgrades at the ISP or systems similar to content distribution networks. A non-negligible infrastructure investment is necessary for centralized or hierarchical systems, such that they are only viable if market mechanisms in the P2P system ensure sufficient payments to finance the infrastructure provisioning. Such market mechanisms will, for instance, be developed in the MMAPPS project (Market Management for Peer-to-Peer Services, [20]). If no such mechanism are available, decentralized search becomes necessary. Structured approaches like symmetric redundant hierarchies or ordered spaces are efficient and scalable but only applicable if reliable trust mechanisms ensure proper collaboration of peers. Otherwise, random structures appear better suited, but should be enhanced with recursive neighborhood knowledge to ensure scalability.

Closed groups can refer to very different deployment scenarios and, accordingly, require many different search designs. Flooding, which is usually believed to be highly inefficient, can be the best choice for corporate LANs, where bandwidth is comparatively cheap and latency low. The flooding mechanisms should strive to exploit the broadcast nature of the majority of LANs, though, to contact all applicable peers with only one message.

As in the case of the wide community, centralized designs can be efficient, provided the required stable server can be made available. In non-corporate environments this necessitates, as before, market mechanisms to be in place for the financial side, whereas corporates have their own mechanisms in place to allocate cost of central infrastructure. In both corporate and non-corporate environments, a stable central server may not be a viable option even if financing were available: the group may be too small, or there may for other reasons be no single entity that could serve as a stable central server, like in mobile ad-hoc.

Without access to proper central infrastructure, decentralized designs are necessary. If the environment of nodes is sufficiently stable, i.e., node joins and leaves do not occur too frequently, structured topologies like symmetric hierarchies or ordered-spaces are the designs of choice. The maintenance burden becomes too high, however, in more dynamic environments. If in addition to the node dynamics also the content or services to be shared are frequently updated, or many replicas cause coherence problems, the required search information should be kept completely on each node, as it is usually the case in mobile groupware. Otherwise, a dynamic node environment is best met with random structures. Flooding can be applied, or random structures exploiting neighborhood information with or without recursion. The larger the group and the slower the dynamics, the higher the need (for scalability reasons) and the easier and more efficient is the use of designs keeping track of neighborhood information.

Finally, systems that interconnect many closed groups (or even group hierarchies) within a wide community are likely, and are also envisioned within MMAPPS [20]. In this scenario, P2P search and lookup algorithms that support or build on hierarchical group structures are helpful, like adaptations of AGILE [18].

8 Summary, Conclusions, and Future Work

While it is impossible to find the one and optimum solution for search in distributed systems, the best solution for each specific application can be identified. The application-relevant fundamental differences between the possible approaches have been identified based on a design space that distinguishes two basic dimensions of distributed search design: functional and structural. Functionally, the search process has been disaggregated into a series of mappings, keyword search, lookup, and routing, and integrated approaches have been identified, viz keyword lookup, name routing, and keyword or semantic routing. Structurally, for each mapping in the functional design space, 11 design options have been derived, grouped into computational approaches, central tables, completely replicated tables, and distributed tables where table structures and topologies can be aligned, unaligned, or both random. A classification and description of example systems validated the framework.

When choosing or designing a search system for a specific application, trade-offs have to be made. The effects of choosing one design over another should be evaluated with respect to the most relevant requirements and criteria in both dimensions, functional and structural, including, *e.g.*, scalability and efficiency or dynamic re-routing facilities, fault-tolerance and adaptability, or infrastructure independence. The design space provides an appropriate tool and high-level evaluation to do so.

New search system design should start with a prioritized set of specific requirements, a corresponding choice of functional succeeded by structural design options, and conclude with the detailed evaluation of existing approaches within this scope, or the design of new alternatives where necessary.

Based on the design space, fundamentally new search designs for peer-to-peer systems, keyword routing based on symmetric hierarchies or ordered spaces, have been identified. They have been outlined in this document and may lead to significant performance improvements for some applications.

General trends in research on P2P lookup and search have been assessed. A break-down of possible deployment scenarios served as a tool and guideline for fundamental design recommendations. Three major recommendations can be given to the research community. First, keyword routing should be embraced further, even in structured system. Second, DHT systems in general currently appear overrated, as their sub-optimality in many deployment scenarios limits their appli-

cability. Third, hybrid P2P systems with central or hierarchical search are efficient and should be re-emphasized, even though they seem less challenging from an academic viewpoint. This particularly applies in light of the forthcoming introduction of market mechanisms into P2P allowing for infrastructure payments and in light of the proliferation of P2P making router or other intermediate infrastructure updates and extensions a viable option.

Going forward, symmetric redundant hierarchy-based keyword routing will be explored in detail and the corresponding system, SHARK, will be developed. Similarly, non-hierarchical ordered-space-based keyword routing seems promising and should be further investigated. In addition, many discussions will be needed to improve comprehensiveness and accuracy of both, the design space with the classification of existing systems as well as the evaluation and implications of the design space and the trade-offs.

Acknowledgements

This work has been performed partially in the framework of the EU IST project MMAPPS “Market Management of Peer-to-Peer Services” (IST-2001-34201), where the ETH Zürich has been funded by the Swiss Bundesministerium für Bildung und Wissenschaft BBW, Bern under Grant No. 00.0275. Special thanks go to Matthias Bossardt for encouraging work on a design space. Additionally, both authors like to acknowledge discussions with all of their project partners.

References

[1] H. Balakrishnan, M. Kaashoek, D. Karger, R. Morris, I. Stoica: *Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications;* ACM SIGCOMM, San Diego, California, U.S.A., August 2001.

[2] M. Balazinska, H. Balakrishnan, D. Karger: *INS/Twine: A Scalable Peer-to-Peer Architecture for Intentional Resource Discovery;* Pervasive 2002 - International Conference on Pervasive Computing, Zurich, Switzerland, August 2002.

[3] P. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, I. Zaihrayeu: *Data Management for Peer-to-Peer Computing: A Vision;* http://sra.itc.it/people/serafini/distribution/webdb-02.pdf, August 2002.

[4] O. Bertsekas, R. Gallager: *Data Networks, 2nd Edition;* Prentice-Hall, U.S.A., 1992.

[5] B. Bloom: *Space/time trade-offs in hash coding with allowable errors;* Communications of the ACM 13, July 1970, pp. 422-426.

[6] Clip2: *The Gnutella Protocol Specification v0.4;* http://www.clip2.com/GnutellaProtocol04.pdf, May 2002.

[7] A. Crespo, H. Garcia-Molina: *Routing Indices For Peer-to-Peer Systems;* International Conference on Distributed Computing Systems (ICDCS), Vienna, Austria, July 2002.

[8] F. Dabek, M. Kaashoek, D. Karger, R. Morris, I. Stoica: *Wide-area cooperative storage with CFS*;18th ACM Symposium on Operating Systems Principles (SOSP '01), Chateau Lake Louise, Banff, Canada, October 2001.

[9] Druschel, Rowstron: *Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems;* IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, 2001, pp.329-350.

[10] L. Ford, D. Fulkerson: *Flows in Networks;* Princeton University Press, Princeton, N.J., U.S.A., 1962.

[11] M. Gritter, D. Cheriton: *An Architecture for Content Routing Support in the Internet*; 3rd Usenix Symposium on Internet Technologies and Systems (USITS), San Francisco, California, U.S.A., March 2001, pp. 37-48.

[12] Groove: *Groove Architecture Flash Movie*; http://www.groove.net/flash/architecture.exe (September 11, 2002).

[13] Z. Haas, M. Pearlman, P. Samar: *Zone Routing Protocol (ZRP);* IETF Internet Draft, draft-ietf-manet-zrp-04.txt, July 2002.

[14] C. Hedrick: *Routing Information Protocol;* Internet Request For Comments RFC 1058, June 1988.

[15] D. Johnson, D. Maltz: *Dynamic Source Routing in Ad Hoc Wireless Networks;* In: Mobile Computing, Vol. 353, Kluwer Academic Publishers, U.S.A., edited by Imielinski and Korth, 1996.

[16] Q. Lv, P. Cao, E. Cohen, K. Li, S. Shenker: *Search and replication in Unstructured Peer-to-Peer Networks;*16th ACM International Conference on Supercomputing (ICS'02), New York, U.S.A., June 2002.

[17] N. Minar: *Distributed Systems Topologies: Part 2* http://www.openp2p.com/pub/a/p2p/2002/01/08/p2p_topologies_pt2.html, June 2002.

[18] J. Mischke, B. Stiller: *Peer-to-peer Overlay Network Management Through AGILE;* IEEE International Symposium on Integrated Network Management (IM), Colorado Springs, CO, U.S.A., March 24-28, 2003.

[19] Computer Engineering and Networks Laboratory (TIK), ETH Zürich, Switzerland, TIK-Report No. 149, August 2002.

[20] MMAPPS*: Annex 1 - Description of Work*; Information Societies Technology (IST) Program, EU Fifth Framework Project, Project Number: IST-2001-34201, 2002.

[21] MMAPPS: *Deliverable 4: Peer-to-peer Services Architecture V2.7*; Information Societies Technology (IST) Program, EU Fifth Framework Project, Project Number: IST-2001-34201, June 2002.

[22] A. Oram (ed.): *Peer-To-Peer: Harnessing the Power of Disruptive Technologies;* O'Reilly&Associates, Sebastopol, U.S.A., 2001.

[23] M. Ott, L. French, R. Mago, D. Makwana, D. Reininger: *Semantic Multicast based on XML Routing;* Submitted to Workshop on Hot Topics in Networks, October 28-29, 2002, Princeton, New Jersey, U.S.A.

[24] C. Perkins, E. Royer: *Ad-hoc on-demand distance vector routing;* MILCOM '97 Panel on Ad Hoc Networks, Monterey, California, U.S.A., November 2-5, 1997.

[25] L. Peterson, B. Davie: *Computer Networks: A Systems Approach, 2. Edition;* Morgan Kaufman Publishers, San Francisco, U.S.A., 2000.

[26] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker: *A Scalable Content-Addressable Network;* ACM SIGCOMM, San Diego, California, U.S.A., August 2001.

[27] Y. Rekhter: *A Border Gateway Protocol 4 (BGP-4);* Internet Request For Comments RFC 1771, March 1995.

[28] *Resource Description Framework (RDF)*; August 2002, http://www.w3.org/RDF (August 29, 2002).

[29] S. Rhea, J. Kubiatowicz: *Probabilistic Location and Routing;* 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), New York, U.S.A., June 2002.

[30] J. Ritter: *Why Gnutella Can't Scale. No, Really.*; http://www.darkridge.com/~jpr5/doc/gnutella.html, (August 23, 2002).

[31] C. Rohrs: *Query Routing for the Gnutella Network, Version 1.0;* May 2002, http://www.limewire.com/developer/query_routing/keyword%20routing.htm (September 3, 2002).

[32] S. Saroiu, P. Gummadi, S. Gribble: *A Measurement Study of Peer-to-peer File Sharing Systems*;Technical Report # UW-CSE-01-06-02, Department of Computer Science & Engineering, University of Washington, Seattle, U.S.A., 2002.

[33] M. Schlosser, M. Sintek, S. Decker, W. Nejdl: *HyperCuP - Hypercubes, Ontologies and Efficient Search on P2P Networks;* International Workshop on Agents and Peer-to-Peer Computing (AP2PC), Bologna, Italy, June 2002.

[34] *The Napster Protocol;* April 2000, http://opennap.sourceforge.net/napster.txt (September 3, 2002).

[35] R. Schollmeier, I. Gruber, M. Finkenzeller: *Routing in Mobile Ad Hoc and Peer-to-Peer Networks - A Comparison;* International Workshop on Peer-to-Peer Computing, Pisa, Italy, May 2002.

[36] Semandex: *What is Content-Based Routing?;* Semandex Information Note, http://www.semandex.net/library.html, August 2002.

[37] Semandex: *Information Services on a Semantic Network;* Semandex Information Note, http://www.semandex.net/library.html, August 2002.

[38] B. Silaghi, S. Bhattacharjee, P. Keleher: *Routing in the TerraDir Directory Service;* SPIE ITCOM'02, Boston, MA, U.S.A., July 2002.

[39] S. Thadani: *Meta Information Searches on the Gnutella Network;* http://www.limewire.com/index.jsp/metainfo_searches, September 2002.

[40] S. Thadani: *Meta Data Searches on the Gnutella Network (Addendum);* http://www.limewire.com/developer/MetaProposal2.htm, September 2002.

[41] K. Truelove, A. Chasin: *Morpheus Out of the Underworld*; July 2002, http://www.openp2p.com/pub/a/p2p/2001/07/02/morpheus.html (August 29, 2002).

[42] B. Zhao, J. Kubiatowicz, A. Joseph: *Tapestry: An infrastructure for fault-tolerant wide-area location and routing;* Technical Report UCB/CSB-01-1141, Computer Science Division, U.C. Berkeley, California, U.S.A., April 2001.

[43] B. Zhao, Y. Duan, L. Huang, A. Joseph, J. Kubiatowicz: *Brocade: Landmark Routing on Overlay Networks;* First International Workshop on Peer-to-Peer Systems (IPTPS), Cambridge, Massachusetts, U.S.A., 2002.