Peer-to-Peer Information Retrieval: An Overview

ALMER S. TIGELAAR, DJOERD HIEMSTRA and DOLF TRIESCHNIGG,
University of Twente

Peer-to-peer technology is widely used for file sharing. In the past decade a number of prototype peer-to-peer information retrieval systems have been developed. Unfortunately, none of these has seen widespread real-world adoption and thus, in contrast with file sharing, information retrieval is still dominated by centralised solutions. In this article we provide an overview of the key challenges for peer-to-peer information retrieval and the work done so far. We want to stimulate and inspire further research to overcome these challenges. This will open the door to the development and large-scale deployment of real-world peer-to-peer information retrieval systems that rival existing centralised client-server solutions in terms of scalability, performance, user satisfaction and freedom.

Categories and Subject Descriptors: H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*Indexing methods*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Search process, selection process*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Distributed systems*

General Terms: Algorithms, Design, Performance, Reliability

ACM Reference Format:
Tigelaar, A.S., Hiemstra, D. and Trieschnigg, D. 2012. Peer-to-Peer Information Retrieval: An Overview. ACM Trans. Inf. Syst. 30, 2, Article 9 (May 2012), 34 pages.
DOI = 10.1145/2180868.2180871 http://doi.acm.org/10.1145/2180868.2180871

1. INTRODUCTION

In centralised search a single party provides a search service over a collection of documents. A few commercial search engines dominate search in the world's largest document collection: the Internet. Their search services use many machines in server farms which *they* exclusively *control*. This raises at least three ethical concerns. Firstly, the search engine operators control the visible information establishing an information monopoly and censorship capabilities. Secondly, conflicts of interest may occur particularly with respect to products and services of competitors. Thirdly, the elaborate tracking of user behaviour forms a privacy risk. Besides this, the main technical concerns are whether such centralised solutions can keep up with the exponential growth of Internet content and the proliferation of dynamic content behind Webforms. We think it would be better if no single party dominates Internet search. We believe users and creators of Web content should collectively provide a search service. This would restore their control over what information they wish to share as well as how they share it. Importantly: no single party would dominate in such a system eliminating the ethical drawbacks of centralised search. Additionally, this enables handling dynamic content and provides scalability, removing the technical weaknesses of centralised systems.

The authors gratefully acknowledge the support of the Netherlands Organisation for Scientific Research (NWO) under project DIRKA (NWO-Vidi), Number 639.022.809.
Author's addresses: A. S. Tigelaar, D. Hiemstra and D. Trieschnigg, Database Group, Faculty of Electrical Engineering and Computer Science, University of Twente, The Netherlands; email: a.s.tigelaar@utwente.nl, d.hiemstra@utwente.nl, and d.trieschnigg@utwente.nl.

ACM Transactions on Information Systems Volume 30, Issue 2, Article 9 (May 2012).
DOI 10.1145/2180868.2180871 http://doi.acm.org/10.1145/2180868.2180871

Unfortunately, no mature solution for this exists. However, peer-to-peer information retrieval could form the foundation for such a collective search platform.

Peer-to-peer architectures provide an alternative to the client-server paradigm that permeates many Internet applications like e-mail, Web browsing and newsgroups. Peers typically have a high degree of both autonomy and volatility. This provides a natural way to distribute processing load and network bandwidth among the participating peers. A peer that joins the network does not only use resources, but also *contributes* resources back. Hence, a peer-to-peer network can potentially scale beyond what is possible in client-server set-ups. However, the price to pay for this is higher algorithmic complexity, security problems and vulnerability to abuse [Aberer and Hauswirth 2002]. Despite this, peer-to-peer networks are widely used for large-scale data sharing, content distribution and application-level multicast [Lua et al. 2005]. In this paper we focus specifically on using peer-to-peer networks for the purpose of information retrieval providing an overview of the work done so far as well as identifying the key challenges in the field.

The rest of this paper is organised as follows: in Section 2 we define what a peer-to-peer system is; provide an overview of commonly used architectures and their characteristics; and identify challenges for peer-to-peer systems. In Section 3 we highlight both the differences and similarities between the most successful application of peer-to-peer technology to date: file sharing, the application that is the subject of this paper: information retrieval, and the closely related field of federated information retrieval. In Section 4 we provide an overview of commonly used optimisation techniques in peer-to-peer information retrieval, and Section 5 contains descriptions of a selection of existing systems. We discuss challenges and key focus areas for future research which will enable better peer-to-peer information retrieval solutions in Section 6. Finally, Section 7 closes the paper.

2. PEER-TO-PEER NETWORKS

2.1. Introduction

A node is a computer connected to a network. This network facilitates communication between the connected nodes through various protocols enabling many distributed applications. The Internet is the largest contemporary computer network with a prolific ecosystem of network applications. Communication occurs at various levels called *layers*. The lowest layers are close to the physical hardware, whereas the highest layers are close to the software. The top layer is the application layer in which communication commonly takes place according to the *client-server paradigm*: *server* nodes provide a resource, while *client* nodes use this resource. An extension to this is the *peer-to-peer paradigm*: here each node is equal and therefore called a *peer*. Each peer could be said to be a client and a server at the same time and thus can both *supply* and *consume* resources. In this paradigm, peers need to cooperate with each other, balancing their mutual resources in order to complete application specific *tasks*. For communication with each other, during task execution, the peers temporarily form *overlay networks*: smaller networks within the much larger network that they are part of. Each peer is connected to a limited number of other peers: its *neighbours*. Peers conventionally transmit data by forwarding from one peer to the next or by directly contacting other, non-neighbouring, peers using routing tables. The *architecture* of a peer-to-peer network is determined by the shape of its overlay network(s), the placement and scope of indices and the protocols used for communication. The choice of architecture influences how the network can be utilised for various tasks such as searching and downloading.

In practice the machines that participate in peer-to-peer networks are predominantly found at the edge of the network, meaning: they are not machines in the big

server farms, but computers in people's homes [Kurose and Ross 2003]. Because of this, a peer-to-peer network typically consists of thousands of low-cost machines all with different processing and storage capacities as well as different link speeds. Such a network can provide many interesting applications, like: file sharing, streaming media and distributed search. Peer-to-peer networks have several properties that make them attractive for these tasks. They usually have no centralised directory or control point and thus also no *central point of failure*. This makes them *self-organizing*, meaning that they automatically adapt when peers join the network, depart from it or fail. The communication between peers uses a common language and is *symmetric* as is the provision of services. This symmetry makes a peer-to-peer network *self-scaling*: each peer that joins the network adds to the available total capacity [Bawa et al. 2003; Risson and Moors 2006].

In the following sections we will first discuss some common applications of peer-to-peer networks and the challenges for such networks, followed by an in-depth overview of commonly used peer-to-peer network architectures.

2.2. Applications

Many applications use peer-to-peer technology. Some examples:

— *Content Distribution*: Usenet, Akamai, Steam.
— *File Sharing*: Napster, Kazaa, Gnutella, BitTorrent.
— *Information Retrieval*: Sixearch, YaCy.
— *Instant Messaging*: ICQ, MSN.
— *Streaming Media*: Tribler, Spotify.
— *Telephony*: Skype, SIP.

Significant differences exist among these applications. One can roughly distinguish between applications with mostly *private data*: instant messaging, and telephony; and *public data*: content distribution, file sharing, information retrieval and streaming media. The term peer-to-peer is conventionally used for this latter category of applications where the sharing of public data is the goal which is also the focus of this article. The interesting characteristic of public data is that there are initially only a few peers that supply the data and there are many peers that demand a copy of it. This asymmetry can be exploited to widely replicate data and provide better servicing for future requests. Since file sharing networks are the most pervasive peer-to-peer application, we will frequently use it as an example and basis for comparison especially in this section which focuses on the common characteristics of peer-to-peer computing. However, in Section 3 we will shift focus to the differences and give a definition of peer-to-peer information retrieval and what sets it apart from other applications.

The concepts *query*, *document*, and *index* will often be used in this article. What is considered to be a query and a document, and what is stored in the index, depends on the application. For most content distribution, file sharing and streaming media systems the documents can be files of all types. The index consists of metadata about those files and the queries are restricted to searching in this metadata space. Information retrieval usually involves a large collection of text documents of which the actual content is indexed and searchable by using free text queries. For searching in instant messaging networks, and telephony applications, the documents are user profiles of which some fields are used to form an index, the query is restricted to searching in one of these fields, for example: 'nickname'.

2.3. Challenges

There are many important challenges specific to peer-to-peer networks [Daswani et al. 2003; Triantafillou et al. 2003]:

— *How to make efficient use of resources?*
Resources are bandwidth, processing power and storage. The higher the efficiency the more requests a system can handle and the lower the costs for handling each request. Peers may vary wildly in their available resources. This heterogeneity raises unique challenges.
— *How to provide acceptable quality of service?*
Measurable important aspects are: low latency, and sufficient, high-quality results.
— *How to guarantee robustness?*
Provide a stable service to peers and the ability to recover from data corruption and communication errors whatever the cause.
— *How to ensure data remains available?*
When a peer leaves the network its content is, temporarily, not accessible. Hence, a peer-to-peer network should engage in quick distribution of popular data to ensure it remains available for as long as there is demand for it.
— *How to provide anonymity?*
The owners and users of peers in the network should have at least some level of anonymity depending on the application. This enables censorship resistance, freedom of speech without the fear of persecution and privacy protection.

Additionally, several behaviours of peers must be handled:

— *Churn*
The stress caused on a network by the constant joining and leaving of peers is termed churn. Most peers remain connected to the network only for a short time. Especially if the network needs to maintain global information, as in a network with a decentralised global index, this can lead to constant costly shifting and rebalancing of data over the network. This behaviour also reduces the availability of data. Peers may leave willingly, but they can also simply crash [Klampanos et al. 2005]. A peer-to-peer network should minimise the communication needed when a peer leaves or joins the network [Stutzbach and Rejaie 2006].
— *Free riding*
A peer-to-peer network is built around the assumption that all the peers in the network contribute a part of their processing power and available bandwidth. Unfortunately, most networks also contain peers that only use the resources of other peers without contributing anything back. These peers are said to engage in free riding. A peer-to-peer network should both discourage free riding and minimise the impact that free riders have on the performance of the network as a whole [Krishnan et al. 2002].
— *Malicious behaviour*
While free riding is just unfair consumption of resources, actual malicious behaviour intends to actively frustrate the usage of resources, either by executing attacks or 'poisoning' the network with fake or corrupted data. A peer-to-peer network should be resilient to such attacks and have mechanisms to detect and remove poisoned data [Kamvar et al. 2003].

Finally, it remains difficult to evaluate and compare different peer-to-peer systems. For this we define the following research challenges:

— *Simulation*
The vast majority of peer-to-peer papers use self-developed simulation frameworks.

This may be surprising since several peer-to-peer simulators exist. However, these have a number of problems like limited ways in which statistics can be obtained, poor documentation and being generally hard to use [Naicken et al. 2006; Naicken et al. 2007]. Creating a framework that can actually be used to conduct experiments for a wide range of peer-to-peer applications is a challenge.

— *Standardised test sets*
Simulations should use standardised test sets so that results of different approaches to peer-to-peer problems can be compared. For a file sharing network this could be a set of reference files of different types like text and video, for an information retrieval network a set of documents, queries and relevance judgements. Creating such test collections is often difficult and labour-intensive. However, they are indispensable for the scientific process.

2.4. Tasks

We distinguish three tasks that every peer-to-peer network performs:

(1) *Searching*: Given a query return some list of document references.
(2) *Locating*: Resolve a document reference to concrete locations from which the full document can be obtained.
(3) *Transferring*: Actually download the document.

From a user perspective the first step is about identifying what one wants, the second about working out where it is and the third about obtaining it [Joseph 2002]. Peer-to-peer networks do not always decentralise all of these tasks and not every peer-to-peer architecture caters well to each task as we will see later. The key point to understand is that searching is different from locating. We will concretely illustrate this difference using three examples.

Firstly, in an instant messaging application searching would be looking for users that have a certain first name or that live in a specific city, for example for all people named Zefram Cochrane in Bozeman, Montana. This search would yield a list with various properties of matching users, including a unique identifier, from which the searcher picks one, for example: the one with identifier 'Z2032'. The instant messaging application can use this to locate that particular user: resolving the identifier to the current machine address of the user, for example: 5.4.20.63. Finally, the transfer step would be sending an instant message to that machine.

Secondly, in information retrieval the search step would be looking for documents that contain a particular phrase, for example 'pizza baking robots'. This would yield a list of documents that either contain the exact phrase or parts thereof. The searcher then selects a document of interest with a unique identifier. Locating would involve finding all peers that share the document with that identifier and finally downloading the document from one of these.

As a final example let us consider the first two tasks in file sharing networks. Firstly, *searching*: given a query find *some possible* files to download. This step yields unique file identifiers necessary for the next step, commonly a hash derived from the file content. Secondly, *locating*: given a specific file identifier find me other peers that offer *exactly* that file. What distinguishes these is that in the first one still has to choose what one wants to download from the search results, whereas in the second one knows exactly what one wants already and one is simply looking for replicas. These two tasks are cleanly split in, for example, BitTorrent [Cohen 2003]. A free text search yields a list of possible torrent files: small metadata files that each describe the real downloadable file with hash values for blocks of the file. This is followed by locating peers that offer parts of this real file using a centralised machine called the tracker. Finally, the download proceeds by obtaining parts of the file from different peers. BitTorrent

thus only decentralises the transfer task, and uses centralised indices for both searching and locating. However, both BitTorrent extensions and many other file sharing networks increasingly perform locating within the peer-to-peer network using a distributed global index. A distributed global index can also be used for the search task. Networks that use aggregated local indices, like Gnutella2, often integrate the search and locate tasks: a free-text search directly yields search results with, for each file, a list of peers from which it can be obtained.

2.5. Architectures

There are multiple possible architectures for a peer-to-peer network. The choice for one of these affects *how* the network can be searched. To be able to search, one requires an index and a way to match queries against entries in this index. Although we will use a number of examples, it is important to realise that what the index is used for is application-specific. This could be mapping filenames to concrete locations in the case of file sharing, user identifiers to machine addresses for instant messaging networks, or terms to documents in the case of information retrieval. In all cases the challenge is that of keeping the latency low whilst retaining the beneficial properties of peer-to-peer networks like self-organisation and load balancing [Daswani et al. 2003]. Based on this there are several subtasks for searching that all affect the latency:

— *Indexing: Who constructs and updates the index? Where is it stored and what are the costs of mutating it?*
  The peers involved in data placement have more processing overhead than others. There can be one big global index, or each peer can index its own content. Peers can specialise in only providing storage space or only filling the index, or they can do both. Where the index is stored also affects query routing.
— *Querying Routing: Along what path is a query sent from an issuing peer to a peer that is capable of answering the query via its index?*
  Long paths are expensive in terms of latency, and slow network links and machines worsen this. The topology of the overlay network restricts the possible paths.
— *Query Processing: Which peer performs the actual query processing (generating results for a specific query based on an index)?*
  Having more peers involved in query processing increases the latency and makes fusing the results more difficult. However, if less peers are involved it is likely that relevant results will be missed.

These search subtasks are relevant to tasks performed in all peer-to-peer networks. In the following paragraphs we discuss how these subtasks are performed in four commonly used peer-to-peer architectures using file sharing as example, since many techniques used in peer-to-peer information retrieval are adapted from file sharing networks.

2.5.1. Centralised Global Index. Early file sharing systems used a *centralised global index* located at a dedicated party, usually a server farm, which kept track of what file was located at which peer in the network. When peers joined the network they sent a list of metadata on files they wanted to share containing, for example, filenames, to the central party which would then include them in its central index. All queries that originated from the peers were directly routed to and processed by that central party. Hence, indexing and searching itself was completely centralised and followed the client-server paradigm. Actually obtaining files, or parts of files, was decentralised by downloading from peers directly. This is sometimes referred to as a brokered architecture, since the central party acts as a mediator between peers. The most famous example of this type of network is Napster. This approach avoids many problems of

other peer-to-peer systems regarding query routing and index placement. However, it has at least two significant drawbacks. Firstly, a central party limits the scalability of the system. Secondly, and more importantly, this central party forms a single point of technical, and legal, failure [Aberer and Hauswirth 2002; Risson and Moors 2006].

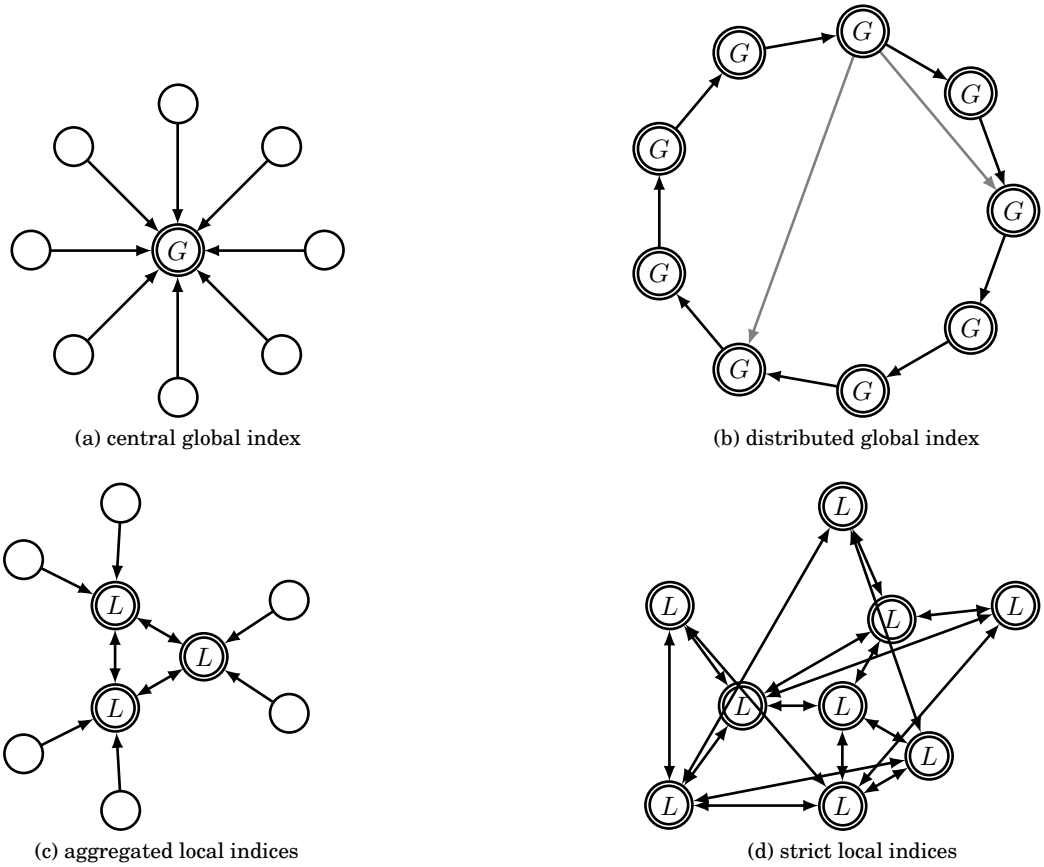
2.5.2. Distributed Global Index. Later systems used a *distributed global index* by partitioning the index over the peers: both the index and the data are distributed in such networks. These indices conventionally take the form of a large key-value store: a distributed hash table [Stoica et al. 2001]. When a peer joins the network it puts the names of the files it wants to share as keys in the global index, and adds its own address as value for these filenames. Other peers looking for a specific file can then obtain a list of peers that offer that file by consulting the global distributed index. Each peer stores some part of this index. The key space is typically divided in some fashion over peers making each peer responsible for keys within a certain range. This also determines the position of a peer in the overlay network. For example: if all peers are arranged in a ring, newly joining peers would bootstrap themselves in between two existing peers and take over responsibility for a part of the key space of the two peers. Given a key, the peer-to-peer network can quickly determine what peer in the network stores the associated value. This key-based routing has its origins in the academic world and was first pioneered in Freenet [Clarke et al. 2001]. There are many ways in which a hash table can topologically be distributed over the peers. However, all of these approaches have a similar complexity for lookups: typically $O(\log n)$, where n is the total number of peers in the network. A notable exception to this are hash tables that replicate all the globally known key-value mappings on each peer. These single-hop distributed hash tables have a complexity of $O(1)$ [Monnerat and Amorim 2009]. The primary difference between hash table architectures is the way in which they adapt when peers join or leave the network and in how they offer reliability and load balancing. A complete discussion of this is beyond the scope of this article, but can be found in [Lua et al. 2005]. A global index can also be implemented using gossip to replicate the full index for the entire network at each peer as done by [Cuenca-Acuna et al. 2003]. However, this approach is not often used and conceptually quite different from hash tables. A key difference is that each peer may have a slightly different view of what the global index contains at a given point in time, since it takes a while for gossip to propagate. In that way it is also close to aggregation. We propose to use the term *replicated global index* to distinguish this approach.

2.5.3. Strict Local Indices. An alternative is to use *strict local indices*. Peers join the network by contacting bootstrap peers and connecting directly to them or to peers suggested by those bootstrap peers until reaching some neighbour connectivity threshold. A peer simply indexes its local files and waits for queries to arrive from neighbouring peers. An example of this type of network is the first version of Gnutella [Aberer and Hauswirth 2002]. This network performs search by propagating a query from its originating peer via the neighbours until reaching a fixed number of hops, a fixed time-to-live, or after obtaining a minimum number of search results [Kurose and Ross 2003]: query flooding. One can imagine this as a ripple that originates from the peer that issued the query: a breadth-first search [Zeinalipour-Yazti et al. 2004]. Unfortunately, this approach scales poorly as a single query generates massive amounts of traffic even in a moderate size peer-to-peer network [Risson and Moors 2006]. Thus, there have been many attempts to improve this basic flooding approach. For example: by forwarding queries to a limited set of neighbours, resulting in a random walk [Lv et al. 2002], by directing the search [Adamic et al. 2001; Zeinalipour-Yazti et al. 2004], or by clustering peers by content [Crespo and Garcia-Molina 2004] or interest [Sripanidkulchai et al. 2003]. An important advantage of this type of network is that no

index information ever needs to be exchanged or synchronised. Thus, index mutations are cheap, and all query processing is local and can thus employ advanced techniques that may be collection-specific, but query routing is more costly than in any other architecture discussed as it involves contacting a large subset of peers. While the impact of churn on these networks is lower than for global indices, poorly replicated, unpopular, data may become unavailable due to the practical limit on the search horizon. Also, peers with low bandwidth or processing capacity can become a serious bottleneck in these networks [Lu 2007].

2.5.4. Aggregated Local Indices. A variation, or rather optimisation, on the usage of local indices are *aggregated local indices*. Networks that use this approach have at least two, and sometimes more, classes of peers: those with high bandwidth and processing capacity are designated as *super peers*, the remaining 'leaf' peers are each assigned to one or more super peers when they join the network. A super peer holds the index of both its own content as well as an aggregation of the indices of all its leafs. This architecture introduces a hierarchy among peers and by doing so takes advantage of their inherent heterogeneity. It was used by FastTrack and in recent versions of Gnutella. Searching proceeds in the same way as when using strict local indices. However, only the super peers participate in routing queries. Since these peers are faster and well connected, this yields better performance compared to local indices, lower susceptibility to bottlenecks, and similar resilience to churn. However, this comes at the cost of more overhead for exchanging index information between leaf peers and super peers [Yang et al. 2006; Lu and Callan 2006]. The distinction between leaf and super peers need not be binary, but can instead be gradual based on, for example, node uptime. Usually leaf peers generate the actual search results for queries using their local index. However, it is possible to even delegate that task to the super peer. The leafs then only transmit index information to the super peer and pose queries.

2.5.5. Discussion. Figure 1 depicts the formed overlay networks for the described peer-to-peer architectures. These graphs serve only to get a general impression of what form the overlay networks can take. The number of participating peers in a real network is typically much higher. Figure 1a shows a centralised global index: all peers have to contact one dedicated machine, or group thereof, for lookups. Figure 1b shows one possible instantiation of a distributed global index shaped like a ring [Stoica et al. 2001]. There are many other possible topological arrangements for a distributed global index overlay, the choice of which only mildly influences the typical performance of the network as a whole [Lua et al. 2005]. These arrangements all share the property that they form *regular graphs*: there are no loops, all paths are of equal length and all nodes have the same degree. This contrasts with the topology for aggregated local indices shown in Figure 1c, which ideally takes the form of a *small world graph*: this has loops, random path lengths, and variable degrees which result in the forming of clusters. Small world graphs exhibit a short global separation in terms of hops between peers. This desirable property enables decentralised algorithms which use only local information for finding short paths. Finally, strict local indices, Figure 1d, either take the form of a small world graph or a random graph depending on whether they include some type of node clustering. A *random graph* can have loops and both random path lengths and node degrees [Aberer and Hauswirth 2002; Kleinberg 2006; Girdzijauskas et al. 2011]. Besides the overall shape of the graph, the path lengths between peers are also of interest. Networks with *interest-based locality* have a short path length between each peer and peers with content similar to its interests. Keeping data closer to peers more likely to request them reduces the latency and overall network load. *Content-based locality* makes finding the majority of relevant contents efficient since they are mostly near to one another: clustering peers with similar content [Lu 2007].



(a) central global index

(b) distributed global index

(c) aggregated local indices

(d) strict local indices

Fig. 1: Overview of peer-to-peer index and search overlays. Each circle represents a peer in the network. Peers with double borders are involved in storing index information and processing queries. A *G* symbol indicates a peer stores a part of a global index, whereas an *L* symbol indicates a local index. The arrows indicate the origin of queries and the directions in which they flow through the system.

Table I shows characteristics of the discussed peer-to-peer architectures and Table II shows an architectural classification for the search task in several existing popular peer-to-peer file sharing networks. We distinguish several groups and types of peers. Firstly, the central peer indicates the machine(s) that store the index in a centralised global index. Secondly, the super peers function as mediators in some architectures. Thirdly, all the peers in the network as a whole and on an individual basis. These distinctions are important since in most architectures the peers involved in constructing the index are not the same as those involved in storage leading to differences in mutation costs. The peer from which a query originates rarely also provides results for that query. Hence, the network needs to route queries from the origin peer to result bearing peers. Queries can be routed either via forwarding between peers or by directly contacting a peer capable of providing results. Even the discussed distributed hash tables use forwarding between peers to 'hop' the query message through intermediate peers in the topology and close in on the peer that holds the value for a particular key. For

Table I: Characteristics of Classes of Peer-to-Peer Networks

	Global Index		Local Indices	
	Centralised	*Distributed*	*Aggregated*	*Strict*
Index				
- Construction	Central Peer	All Peers	All Peers	All Peers
- Storage	Central Peer	All Peers (Shared)	Super Peers	All Peers (Indiv.)
- Mutation Cost*	Low	High	Low	None
Query Routing				
- Method	Direct	Forwarding	Forwarding	Forwarding
- Parties	Central Peer	Intermediate Peers	Super Peers	Neighbour Peers
- Complexity	$O(1)$	$O(\log N)^\dagger$	$O(N_s - 1)^\ddagger$	$O(N - 1)$
Query Processing				
- Peer Subset	Central Only	Small	Medium	Large
- Latency	Low	Medium	Medium	High
- Result Set Unit	Query	Term	Query	Query
- Result Fusion	–	Intersect	Merge	Merge
- Exhaustive	Yes	Yes	No$^\diamond$	No$^\diamond$

This list is not exhaustive, but highlights latency aspects of these general architectures important for information retrieval.
*In terms of network latency and bandwidth usage from [Yang et al. 2006].
$^\dagger O(1)$ distributed hash tables also exist [Monnerat and Amorim 2009; Risson and Moors 2006].
‡Applies to the number of super peers N_s.
$^\diamond$Searches are restricted to a subset of peers and thus to a subset of the index.

all architectures the costs of routing a query is a function of the size of the network. However, the number of peers that perform actual processing of the query, and generate search results, varies from a single peer, in the centralised case, to a large subset of peers when using strict local indices. Lower latency can be achieved by involving fewer peers in query processing. For information retrieval networks returned results typically apply to a whole query, except for the distributed global index, that commonly stores results using individual terms as keys. It is necessary to somehow fuse results obtained from different peers except when using a central global index. A distributed global index must intersect the lists of results for each term. Whereas local indices can typically merge incoming results with the list of results obtained so far. The simplest form of merging is appending the results of each peer to one large list.

The discussed approaches have different characteristics regarding locating suitable results for a query. The approaches that use a global index can search exhaustively. Therefore, it is easy to locate results for rare queries in the network: every result can always be found. In contrast, the approaches that use local indices can flood their messages to only a limited number of peers. Hence, they may miss important results and are slow to retrieve rare results. However, obtaining popular, well replicated, results from the network incurs significantly less overhead. Additionally, they are also more resilient to churn, since there is no global data to rebalance when peers join or leave the system [Lua et al. 2005]. Local indices give the peers a higher degree of autonomy, particularly in the way in which they may shape the overlay network [Daswani et al. 2003]. Advanced processing of queries, such as stemming, decompounding and query expansion, can be done at each peer in the network when using local indices as each peer receives the original query. When using a global index these operations all have to be done by the querying peer, which results in that peer executing multiple queries derived from the original query thereby imposing extra load on the network. Further-

Table II: Classification of Free-text *Search* in Peer-to-Peer File Sharing Networks

	Global Index		Local Indices	
	Centralised	*Distributed*	*Aggregated*	*Strict*
BitTorrent	■			
FastTrack			■	
FreeNet		■		
Gnutella				■
Gnutella2			■	
Kad		■		
Napster	■			

more, one should realise that an index is only part of an information retrieval solution and cannot solve the relevance problem by itself [Zeinalipour-Yazti et al. 2004].

Solutions from different related fields apply to different architectures. Architectures using a global index have more resemblance to cluster and grid computing, whereas those using a local index have most in common with federated information retrieval. Specifically, usage of local indices gives rise to the same challenges as in federated information retrieval: resource description, collection selection and search result merging, as we will discuss later in Section 3.3 [Callan 2000].

An index usually consists of either one or two layers: a *one-step index* or a *two-step index*. In both cases the keys in the index are terms. However, in a one-step index the values are direct references to document identifiers, whereas in a two-step index the values are peer identifiers. Hence, a one-step index requires only one lookup to retrieve all the applicable documents for a particular term. Strict local indices are always one-step. In a two-step index the first lookup yields a list of peers. The second step is contacting one, or more, peers to obtain the actual document identifiers. A one-step index is a straight *document index*, whereas a two-step index actually consists of two layers: a *peer index* and a document index per peer. A network with aggregated local indices is two-step when the leaf peers are involved in generating search results and the aggregated indices contain leaf peer identifiers. Two-step indices are most commonly used in combination with a distributed global index: the global index maps terms to peers that have suitable results for those terms. Note that a distributed global index requires contacting other peers most of the time for index lookups: even if we would store terms as keys and document identifiers as values, to perform a lookup one still needs to hop through the distributed hash table to find the associated value for a key. However, this is conceptually still a one-step index, since the distributed hash table forms one index layer. Note that some clustering approaches use a third indexing layer intended to map queries to topical clusters.

Peer-to-peer networks are conventionally classified as either structured or unstructured. The approach with strict local indices is classified as unstructured and the approach that uses a distributed global index as structured. However, we agree with [Risson and Moors 2006] that this distinction has lost its value. This is because most modern peer-to-peer networks assume *some type* of structure: the strict local indices approach is rarely applied. The two approaches are sometimes misrepresented as competing alternatives [Suel et al. 2003], whereas their paradigms really augment each other. Hence, some systems combine some properties of both [Rosenfeld et al. 2009]. The centralised global index is structured because the central party can be seen as one very powerful peer. However, the overlay networks that form at transfer time are unstructured. Similarly, the aggregated indices approach is sometimes referred to as semistructured since it fits neither the structured nor the unstructured definition. We

believe it is more useful to describe peer-to-peer networks in terms of their specific structure and application and the implications this has for real-world performance. Hence, we will not further use the structured versus unstructured distinction in this article. Rather, we will focus on our primary application: searching in peer-to-peer networks, specifically in the information retrieval context.

3. PEER-TO-PEER INFORMATION RETRIEVAL NETWORKS

3.1. Introduction

In an information retrieval peer-to-peer network the central task is *searching*: given a *query* return some list of document references: the *search results*. A query can originate from any peer in the network, and has to be routed to one or more other peers that can provide search results based on an index. The peers thus supply and consume results. A search result is a compact representation of a document. A document can contain text, image, audio, video or a mixture of these [Zeinalipour-Yazti et al. 2004]. A search result is sometimes called a *snippet* and at least includes a pointer to the full document and commonly additional metadata like a title, a summary, the size of the document, et cetera. A concrete example: search results as displayed by modern search engines. Each displayed result links to the associated full document. The compact representation provides a first filtering opportunity for users enabling them to choose what links they want to follow.

Peer-to-peer information retrieval networks can be divided into two classes based on the location of the documents pointed to. Firstly, those with *internal document* references where the documents have to be downloaded from other peers within the network, for example: digital libraries [Lu and Callan 2006; Di Buccio et al. 2009]. Secondly, those with *external document* references where obtaining the actual documents, locating and transferring, is outside of the scope of the peer-to-peer network, for example: a peer-to-peer Web search engine [Bender et al. 2005b].

In the next sections we compare peer-to-peer information retrieval networks with other applications and paradigms.

3.2. Comparison with File Sharing Networks

File sharing networks are used to search for, locate and download files that users of the peer-to-peer network share. The *searching* in such networks is similar to peer-to-peer information retrieval. A free text query is entered after which a list of files is returned. After searching the user selects a file of interest to download which usually has some type of globally unique identifier, like a content-based hash. The next step is *locating* peers that have a copy of the file. It may then be either *transferred* from one specific peer, or from several peers simultaneously in which case specific parts of the file are requested from each peer and stitched back together after the downloads complete.

The tasks of locating peers and especially transferring content are the primary application of file sharing networks and the focus area of research and performance improvements. Searches in such networks are usually for known items, whereas in information retrieval networks the intent is more varied [Lu 2007]. While some information retrieval networks also provide locating and downloading operations, they typically focus on optimisations for the search task. Besides this general difference in focus, there are at least three concrete differences as well.

Firstly, the search index for file sharing is usually based only on the names of the files available on the network and not on their content as is the case for information retrieval. Such a name index is smaller than a full document index [Suel et al. 2003]. Hence, there are also fewer postings for each term which makes it less costly to perform intersections of posting lists, an operation common in a distributed global index

Table III: Differences between Locating for File Sharing and Searching in Information Retrieval using a Two-Step Index

	File Sharing	Information Retrieval
Application	Locating	Searching
Index		
– Content	File identifiers	Document content
– Size	Small	Large
– Dominant Operation	Append	Update
– Document Location	Internal	External
– First Step Mapping	$fileid \rightarrow \{peer\}$	$term \rightarrow \{peer\}$
– Second Step Mapping	$fileid \rightarrow file$	$term \rightarrow \{document\}$
– Mapping Type	Exact Lookup	Relevance Ranking
– Result Fusion	Trivial	Difficult
Dominant Data Exchange		
– Unit	Files	Search results
– Size	Megabytes+ (large)	Kilobytes (small)
– Emphasis	High throughput	Low latency

[Reynolds and Vahdat 2003]. Because of their small size a centralised index scales well for name indices [Lu 2007]. However, centrally searched networks have become unpopular largely because of legal reasons.

Secondly, when a file is added to a file sharing index it does not change. If an adjusted version is needed, it is simply added as a new file. Hence, index updates are not required. In contrast in an information retrieval network when the underlying document changes, the associated search results generated from that document have to change as well. Hence, the index needs to be updated so that the search results reflect the changes to the document pointed to.

Thirdly, since the emphasis in a file sharing network is on downloading files as fast as possible it is important to have a *high throughput*. In contrast, in information retrieval the search task dominates in which *low latency* is the most important [Reynolds and Vahdat 2003]. More concretely: it is acceptable if the network takes half a minute to locate the fastest peers for a download, whereas taking that long is not acceptable for obtaining quality search results. Table III summarises the differences assuming a two-step index and a peer-to-peer Web search engine for information retrieval. For file sharing the index shown is the one used for *locating* a specific file, whereas for information retrieval it is for *searching*. The first-step mapping is always made at the level of the whole network, whereas the second-step mapping is made at a specific peer.

3.3. Comparison with Federated Information Retrieval

In federated information retrieval[1] there are three parties as depicted in Figure 2: *clients* that pose queries, one *mediator*, and a set of search *servers* that each discloses a collection of documents: resembling strict local indices. The search process begins when a client issues a query to the mediator. The mediator has knowledge of a large number of search servers and contacts an appropriate subset of these for answering the query. Each search server then returns a set of search results for the query. The mediator merges these results into one list and returns this to the client [Callan 2000].

[1]This is also referred to as distributed information retrieval. However, 'distributed' can be confused with general distributed systems such as server farms and grids. Hence, we stick to the now more popular term federated information retrieval.

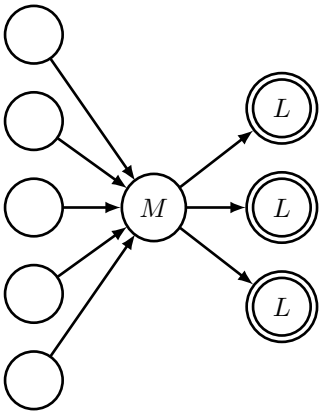


Fig. 2: Schematic depiction of federated information retrieval. Each circle represents a peer in the network, those at the left are clients. Peers with double borders, at the right, are servers that maintain local indices marked with L. In between is the mediator node denoted with an M. The arrows indicate the origin of queries and the direction in which these flow through the system.

Similarities. There are three challenges that form the pillars of federated information retrieval which it has in common with peer-to-peer information retrieval [Callan 2000]. Firstly, there is the *resource description problem*: the mediator either needs to receive from each search server an indication of the queries it can handle [Gravano et al. 1997], in the case of *cooperative* servers, or the mediator needs to find this out by probing the search servers if they are *uncooperative* [Du and Callan 1998; Shokouhi and Zobel 2007]. In either case the end result is a resource description of the search server. These descriptions are typically kept small for efficiency reasons, as even large collections can be described with a relatively small amount of data [Tigelaar and Hiemstra 2010]. The description can consist of, for example: summary statistics, collection size estimates, and/or a representative document sample. In a peer-to-peer information retrieval network the peers need to know to what other peers they can send a query. Hence, resource descriptions are also needed. The advantage of peer-to-peer networks is that peers are cooperative and speak a designed and agreed upon protocol, making exchange of resource descriptions easier. However, peers may have an incentive to cheat about their content, which creates unique challenges specific to peer-to-peer networks.

Secondly, there is the *collection selection problem*: after acquiring resource descriptions the next step is selecting a subset of search servers that can handle the query. When the mediator receives a new query from a client it can quickly score it locally against the acquired resource descriptions to determine the servers most likely to yield relevant search results for the query. The algorithms for determining the best servers in federated information retrieval can be divided in two groups. Firstly, those that treat resource descriptions as big documents without considering individual documents within each resource: CORI, CVV and KL-Divergence based [Callan et al. 1995; Yuwono and Lee 1997; Xu and Croft 1999]. Secondly, those that do consider the individual documents within each resource: GlOSS, DTF, ReDDE [Gravano et al. 1999; Si and Callan 2003a; Nottelmann and Fuhr 2007]. Although considering individual documents gives better results, it also increases the complexity of resource descriptions and the communication costs. Additionally, most existing resource selection algorithms are

designed for use by a single mediator party making them difficult to apply in a network with, for example, aggregated local indices. Resource selection according to the unique characteristics of peer-to-peer networks requires development of new algorithms [Lu 2007].

Thirdly, there is the *result merging problem*: once the mediator has acquired results from several search servers these need to be merged into one coherent list. If all servers would use the same algorithm to rank their results this would be easy. However, this is rarely the case and exact ranking scores are commonly not included. The first step in merging is to normalise the scores globally, so that they are resource independent. In federated information retrieval CORI or the SemiSupervised Learning (SSL) merging algorithm can be used for this [Si and Callan 2003b]. However, in peer-to-peer environments the indexed document collections often vary widely in their sizes which makes CORI unlikely to work well. SSL requires a sample database which makes it undesirable in peer-to-peer networks cautious about bandwidth usage. An alternative approach is to recalculate document scores at the mediator as done by Kirsch's algorithm [Kirsch 1997] which is quite accurate and has low communication costs by only requiring each resource to provide summary statistics. However, this also requires knowledge of global corpus statistics which is costly to obtain in peer-to-peer networks with local indices. Result merging in peer-to-peer information retrieval networks requires an algorithm that can work effectively with minimal additional training data and communication costs, for which none of the existing algorithms directly qualifies. Result merging in existing networks has so far relied on simple frequency-based methods, and has not provided any solution to relevance-based result integration [Lu 2007].

Differences. The first noticeable difference with peer-to-peer information retrieval is the strict specialisation of the various parties. The clients only issue queries whereas the search servers only serve search results. This also determines the shape of the rigid overlay network that forms: a bipartite graph with clients on one side, servers on the other side and the mediator in the middle. Indeed, federated information retrieval is much closer to the conventional client-server paradigm and commonly involves machines that already 'know' each other. This contrasts with peer-to-peer networks where peers take on these roles as needed and frequently interact loosely with 'anonymous' other machines. Additionally, a peer-to-peer network is subject to significant churn, availability and heterogeneity problems which only mildly affect federated information retrieval networks due to the strict separation of concerns [Lu 2007].

A second difference is the presence of the mediator party. To the clients the mediator appears as one entry point and forms a façade: clients are never aware that multiple search servers exist at all. This has the implication that all communication is routed through the mediator which makes it a single point of failure. In practice a mediator can be a server farm to mitigate this. However, it still remains a single point of control, similar to completely centralised search systems, which can create legal and ethical difficulties. A peer-to-peer network with one central 'mediator' point for routing queries is conceptually close to a federated information retrieval network [Lu 2007]. However, most peer-to-peer networks lean towards distributing this mediation task, mapping queries to peers that can provide relevant search results, over multiple peers.

4. EXISTING RESEARCH

4.1. Introduction

Peer-to-peer information retrieval has been an active research area for about a decade. In this section we first reveal the main focus of peer-to-peer information retrieval, followed by an in-depth overview of optimisation techniques developed over the years.

A practical view on the goal of peer-to-peer information retrieval is minimising the number of messages sent per query while maintaining high recall and precision [Zeinalipour-Yazti et al. 2004]. There are several approaches to this which represent trade-offs. Let us start with the two common strategies to partition indices over multiple machines: *partition-by-document* and *partition-by-keyword* [Li et al. 2003]. In partition-by-document each peer is responsible for maintaining a local index over a specific set of documents: the postings for all terms of a particular document are located at one specific peer. In some cases the documents themselves are also stored at that peer, but they need not be. The strict and aggregated local indices architectures are commonly used in peer-to-peer networks that use this partitioning. In contrast, in partition-by-keyword each peer is responsible for storing the postings for some specific keywords in the index. A natural architecture for this is the distributed global index.

An early investigation into the feasibility of a peer-to-peer Web search network was done by [Li et al. 2003]. They view partition-by-document as a more tractable starting point, but show that partition-by-keyword can get within range of the performance of partition-by-document by applying various optimisations to a distributed global index. In contrast [Suel et al. 2003] conclude that partition-by-document approaches scale poorly, because document collections do not 'naturally' cluster in a way that allows query routing to a small fraction of peers and thus each query requires contacting nearly all peers in the system. Perhaps due to this paper much of the research in peer-to-peer information retrieval has focused on partition-by-keyword using a distributed global index [Klampanos and Jose 2004].

Unfortunately, a distributed global index is not without drawbacks since it is intended for performing efficient lookups, not for efficient search [Bawa et al. 2003]. Firstly, a hash table provides load balancing rather naively, by resorting to the uniformity of the hash function used [Triantafillou et al. 2003]. As term posting lists differ in size this can cause hotspots to emerge for popular terms which debalances the load. Secondly, the intersection of term posting lists used in distributed global indices ignores the correlations between terms which can lead to unsatisfactory search accuracy [Lu 2007]. Thirdly, the communication cost for an intersection grows proportionally with the number of query terms and the length of the inverted lists. Several optimisations have been proposed such as storing multiterm query results for a particular term locally to avoid intersections and requiring each peer to store additional information for terms strongly correlated with the terms it already stores. The choice of resource descriptions in a distributed global index is thus limited by the high communication costs of index updates: full-text representations are unlikely to work well due to the massive network traffic that this requires. Fourthly, skewed corpus statistics as a result of term partitioning may lead to globally incomparable ranking scores. Finally, distributed hash tables are vulnerable to various network attacks that compromise the security and privacy of users [Steiner et al. 2007].

Many authors fail to see a number of benefits unique to partition-by-document local indices, such as the low costs for finding popular items, advanced query processing, inexpensive index updates and high churn resilience. Admittedly, the primary challenge for such indices is routing the query to suitable peers. Our stance is that both approaches have their merit and complement each other. Recent research indeed confirms the effectiveness of using local indices for popular query terms and a global index for rare query terms [Rosenfeld et al. 2009].

[Li et al. 2003] conclude that Web-scale search is not possible with peer-to-peer technology. The overhead introduced by communication between peers is too large to offer reasonable query response times given the capacity of the Internet. However, much work, discussed in the next section, has been done since their paper and the nature and capacity of the Internet has changed significantly in the intervening time.

[Yang et al. 2006] compare the performance of several peer-to-peer architectures for information retrieval combined with common optimisations. They test three approaches: a distributed global index augmented with Bloom filters and caching; aggregated local indices with query flooding; and strict local indices using random walks. All of these are one-step term-document indices. Interestingly, they all consume approximately *the same amount* of bandwidth during query processing, although the aggregated local indices are the most efficient. However, the distributed global index offers the lowest latency of these three approaches, closely followed by aggregated local indices and strict local indices being orders of magnitude slower. For all approaches the forwarding of queries in the network introduces the most latency, while answering queries is relatively inexpensive. Even though the distributed global index is really fast its major drawback rears its ugly head at indexing and publishing time. When new documents are added to the network this uses six times as much bandwidth and nearly three times as much time compared to the aggregated local indices for updating the posting lists. Strict local indices resolve all this locally and incur no costs in terms of time or bandwidth for publishing documents. This study clearly shows that an architecture should achieve a balance between *retrieval speed* and *update frequency*.

4.2. Optimisation Techniques

In this section we discuss several optimisation approaches. There are two reasons to use these techniques. One is to reduce bandwidth usage and latency, the other is to improve the quality and quantity of the search results returned. Most techniques discussed influence both of these aspects and offer trade-offs, for example: one could compromise on quantity to save bandwidth and on quality to reduce latency.

4.2.1. Approximate Intersection of Posting Lists with Bloom Filters and Min-Wise Independent Permutations. [Cuenca-Acuna et al. 2003; Reynolds and Vahdat 2003; Suel et al. 2003; Zhang and Suel 2005; Michel et al. 2005a; Michel et al. 2006]

When using a distributed global index a multiterm query requires multiple lookups in the distributed hash table. The posting list for each term needs to be intersected to find the documents that contain all query terms. Exchanging posting lists can be costly in terms of bandwidth, particularly for popular terms with many postings, thus smaller Bloom filters derived from these lists can be transferred instead. Bloom Filters were first used in peer-to-peer information retrieval by [Reynolds and Vahdat 2003].

A Bloom filter is an array of bits. Each bit is initially set to zero. Two operations can be carried out on a Bloom filter: inserting a new value and testing whether an existing value is already in the filter. In both cases k hash functions are first applied to the value. An insert operation, based on the outcome, sets k positions of the Bloom filter to one. Membership tests read the k positions from the Bloom filter. If all of them equal one the value *might be* in the data set. However, if one of the k positions equals zero the value *is certainly not* in the data set. Hence, false positives are possible, but false negatives never occur [Bloom 1970; van Heerde 2010, p. 82]. Bloom filters are an attractive approach for distributed environments because they achieve smaller messages which leads to huge savings in network I/O [Zeinalipour-Yazti et al. 2004]

Consider an example in the peer-to-peer information retrieval context: peer Q poses a query q consisting of terms a and b. We assume that term a has the longest posting list. Peer A holds the postings $P(a)$ for term a, derives a Bloom filter $F(a)$ from this and sends it to peer B that contains the postings $P(b)$ for term b. Peer B can now test the membership of each document in $P(b)$ against the Bloom filter $F(a)$ and send back the intersected list $P(b) \cap F(a)$ to peer Q as final result. Since this can still contain false positives, the intersection can instead by sent back to peer A, which can remove false positives since it has the full postings $P(a)$, the result is then $P(a) \cap (P(b) \cap F(a))$: the

true intersection for terms a and b, which can be sent as result to peer Q. Bandwidth savings occur when sending the small $F(a)$ instead of the large $P(a)$ from peer A to B. However, this approach requires an extra step if one wants to remove the false positives [Reynolds and Vahdat 2003].

False positives are the biggest drawback of Bloom filters: the fewer bits used, the higher the probability a false positive occurs. Large collections require more bits to be represented than smaller ones. Unfortunately, Bloom filters need to have the same size for intersection and union operations. This makes them unsuitable for networks in which the peers have collections that vary widely in the number of stored documents.

Bloom filters can be used to perform approximate intersection of posting lists. However, as a step prior to that it is also interesting to estimate what an additional posting list would do in terms of intersection to the lists already obtained. This task only requires cardinality estimates and not the actual result of an intersection. While Bloom filters can be used for this, several alternatives are explored by [Michel et al. 2006]. The most promising is Min-Wise Independent Permutations (MIPs). This requires a list of numeric document identifiers as input values. Firstly, this method applies k linear hash functions, with a random component, to the values each yielding a new list of values. Secondly, the resulting k lists are all sorted, yielding k permuted lists, and the minimum value of each of these lists is taken and added to a new list: the MIP vector of size k. The fundamental insight is that each element has the same probability of becoming the minimum element under a random permutation. The method estimates the intersection between two MIP vectors by taking the maximum of each position in the two vectors. The number of distinct values in the resulting vector divided by the size of that vector forms an estimate of the overlap between them. The advantage is that even if the input vectors are of unequal length, it is still possible to use only the first few positions to get a, less accurate, approximation. [Michel et al. 2006] show that MIPs are much more accurate than Bloom filters for this type of estimation.

4.2.2. Reducing the Length of Posting Lists with Highly Discriminative Keys. [Skobeltsyn et al. 2009; Luu et al. 2006]

An alternative way of reducing the costs of posting list intersection for a distributed global index is by making the lists themselves shorter. To achieve this instead of building an index over single terms, one can build one over entire multiterm queries. This is the idea behind highly discriminative keys. No longer are all terms posted in a global distributed index, but instead multiterm queries are generated from a document's content that discriminate that document well from others in the collection. The result: more postings in the index, but shorter posting lists. This offers a solution to one of the main drawbacks of using distributed hash tables: intersection of large posting lists.

4.2.3. Limiting the Number of Results to Process with Top k *Approaches.* Processing only a subset of items during the search process can yield performance benefits: less data processing and lower latency. Various algorithms, discussed shortly, can be used to retrieve the top items for a particular query without having to calculate the scores for all the items. Retrieving top items makes sense as it has been shown that users of Web search engines prefer quality over quantity with respect to search results: more precision and less recall [Oulasvirta et al. 2009]. Top k approaches have been applied to various architectures and at various stages in peer-to-peer information retrieval:

— Top k results requesting [Cuenca-Acuna et al. 2003]
  A simple way to optimise the system is to only request the top results. Approaches that use local indices always apply a variable form of limited result requesting implicitly by bounding the number of hops made when flooding or by performing a random walk that terminates. However, that number can also be explicitly set to

a constant by the requester as is done for the globally replicated index used by [Cuenca-Acuna et al. 2003]. They first obtain a list of k search results and keep contacting nodes as long as the chance of them contributing to this top k remains high. The top results stabilise after a few rounds.

— Top k query processing [Suel et al. 2003; Balke et al. 2005; Michel et al. 2005a; Zhang and Suel 2005]
This approach has its roots in the database community, particularly in the work of [Fagin et al. 2001]. Several variations exist, all with the same basic idea: we can determine the top k documents given several input lists without having to examine these lists completely and while not adversely affecting performance. This is often used in cases where a distributed global index is used and posting lists have to be intersected. The *threshold algorithm* is the most popular [Michel et al. 2005a; Suel et al. 2003]. This algorithm maintains two data structures: a queue with peers to contact for obtaining search results and a list with the current top k results. Peers in the queue are processed one by one, each returning a limited set of k search results of the form $(document, score)$ sorted by score in descending order. For a distributed global index these are the top items in the posting list for a particular term. The algorithm tracks two scores for each unique document: worst and best. The worst score is the sum of the scores for a document d found in all result lists in which d appeared. The best score is the worst score plus the lowest score (of some other document) encountered in the result lists in which d did not appear. Since all the result lists are truncated, this last score forms an upper bound of the best possible score that would be achievable for document d. The current top k is formed by the highest scoring documents seen so far based on their worst score. If the best score of a document is lower than the threshold, which is the worst score of the document at position k in the current top k results, it need not be considered for the top k. The algorithm thus bases the final intersection on only the top k results from each peer, which provably yields performance equivalent to 'sequentially' intersecting the entire lists. This thus saves both bandwidth and computational costs without negatively affecting result quality. A drawback is that looking up document scores requires random access to the result lists [Suel et al. 2003]. [Zhang and Suel 2005] later investigated the combination of top k query processing with several optimisation techniques. They draw the important conclusion that different optimisations may be appropriate for queries of different lengths. [Balke et al. 2005] show that top k query processing can also be effective in peer-to-peer networks with aggregated local indices.

— Top k result storing [Tang et al. 2002; Tang and Dwarkadas 2004; Skobeltsyn and Aberer 2006; Skobeltsyn et al. 2007; Skobeltsyn et al. 2009]
One step further is only *storing* the top k results for a query, or term, in the index. [Skobeltsyn and Aberer 2006] take this approach as a means to further reduce traffic consumption. Related to this is the approach of [Tang and Dwarkadas 2004] that store postings only for the top terms in a document. They state that while indexing only these top terms might degrade the quality of search results, it likely does not matter since such documents would not rank high for queries for the other non-top terms they contain anyway.

4.2.4. Reducing the number of Peers involved in Index Lookups by Global Index Replication. [Cuenca-Acuna et al. 2003; Galanis et al. 2003]

Lookups to map queries to peers are expensive when they involve contacting other peers regardless of the architecture used. What if a peer can do all lookups locally? The authors of the PlanetP system explore this novel approach. They essentially replicate a full global index at each peer: a list of all peers, their IP addresses, current net-

work status and their Bloom filters for terms. This information is spread through the network using *gossip*. If something changes at a peer it gossips the change randomly to each of its neighbours until enough neighbouring peers indicate that they already know about the rumour. Each peer that receives rumors also spreads it in the same way. There is the possibility that a peer misses out on a gossip, to cope with this the authors periodically let peers exchange their full directory and they also piggyback information about past rumors on top of new ones. Whilst this is an interesting way to propagate indexing information, it is unfortunately also slow: it takes in the order of hundreds of seconds for a network of several thousand peers to replicate the full index information at each peer. This approach has not seen widespread adoption and is perhaps best suited to networks with a small number of peers due to scalability issues [Zeinalipour-Yazti et al. 2004].

Although we prefer to label this approach as a global index, it can also be viewed as a very extreme form of aggregation where each peer holds aggregate data on every other peer in the network. Note that this approach differs from a single-hop distributed hash-table, since it uses no hashing and no distributed key space. Hence, the topology of the network is not determined by a key space.

4.2.5. Reducing Processing Load by Search Result Caching. [Reynolds and Vahdat 2003; Skobeltsyn and Aberer 2006; Skobeltsyn et al. 2007; Zimmer et al. 2008; Skobeltsyn et al. 2009; Tigelaar and Hiemstra 2011; Tigelaar et al. 2011]

It makes little sense to reconstruct the search result set for the same query over and over again if it does not really change. Performance can be increased significantly by caching search results. [Skobeltsyn and Aberer 2006] use a distributed hash table to keep track of peers that have cached relevant search results for specific terms. Initially this table is empty, and each (multiterm) query is first broadcast through the entire peer-to-peer network, using a shower broadcast with costs $O(n)$ for a network of n peers. After this step the peer that obtained the search results registers itself as caching in the distributed hash table for each term in the query. This allows for query subsumption: returning search results for subsets of the query terms in the absence of a full match. The authors base the content of the index on the queries posed within the network, an approach they term *query-driven indexing*. This significantly reduces network traffic for popular queries while maintaining a global result cache that adapts in real-time to submitted queries.

4.2.6. Reducing the Number of Peers Involved in Query Processing by Clustering. [Bawa et al. 2003; Sripanidkulchai et al. 2003; Crespo and Garcia-Molina 2004; Klampanos and Jose 2004; Akavipat et al. 2006; Klampanos and Jose 2007; Lu and Callan 2007; Lele et al. 2009; Tirado et al. 2010]

When using local indices, keeping peers with similar content close to each other can make query processing more efficient. Instead of sending a query to all peers it can be sent to a cluster of peers that covers the query's topic. This reduces the total number of peers that need to be contacted for a particular query. Unfortunately, content-based clustering does not occur naturally in peer-to-peer networks [Suel et al. 2003]. Hence, [Bawa et al. 2003] organise a peer-to-peer networks by using topic segmentation. They arrange the peers in the network in such a way that only a small subset of peers, that contain matching relevant documents, need to be consulted for a given query. Clustering peers is performed based on either document vectors or full collection vectors. They then use a two-step process to route queries based on the topic they match. They first find the cluster of peers responsible for a specific topic and forward the query there. After this the query is flooded within the topical cluster to obtain matches. They conclude that their architecture provides good performance, both in terms of retrieval quality and in terms of latency and bandwidth. Unfortunately, their system requires a

central directory for initially finding a good topic cluster for query routing. [Akavipat et al. 2006] show how to do clustering without such a central directory.

[Klampanos and Jose 2007] evaluate cluster-based architectures for large-scale peer-to-peer information retrieval focusing on single-pass clustering with both a variable and fixed number of clusters. They find that the predominantly small size of Web documents makes them hard to relate to other documents thereby leading to poor clustering. Clustering mechanisms fail to discover the structure of the underlying document distribution leading to the situation where not enough relevant sources can be contacted to route a query to. This is due to the loss of information inherent in the creation of cluster centroids. They propose two solutions. Firstly, replicating documents part of popular clusters on multiple peers, leading to a significant improvement in effectiveness. Although this does not solve the problem for unpopular topics, it could work sufficiently well for most users. Secondly, assuming a relevance feedback mechanism exists and using this to alter the centroids of the global topic clusters. The weight of each term in a cluster is then determined by the relevance of that cluster to the query based on the feedback. They show the usefulness of both replication and relevance feedback which lead to better query routing and higher precision, further emphasizing relevance feedback as a promising and natural evolution of current peer-to-peer information retrieval technologies.

Interest-based clustering works either by shortening the path lengths between peers with similar interest, meaning: peers which pose similar queries, or by bringing peers with a particular interest closer to peers with matching content. Although not exactly the same, both aim to reduce the number of hops needed for obtaining relevant content. In the first case by leveraging cached information present at peers with similar interests: caches at other consuming peers, while the second case brings one closer to the origin of information: providing peers that contain original content [Sripanidkulchai et al. 2003; Akavipat et al. 2006].

The two clustering approaches: by content and by interest, can also be combined.

4.2.7. Reducing Latency and Improving Recall using Random Walks. [Lv et al. 2002; Yang et al. 2006]

Peer-to-peer systems with local indices are conventionally searched with query flooding. That approach is theoretically exhaustive, but because of tractability it is applied in a non-exhaustive way by bounding the number of hops. [Lv et al. 2002] propose an alternative to this by using random walks. Instead of searching in a breadth-first manner: forwarding the queries to all neighbours, we search depth-first by forwarding the query only to one neighbour. Such a walk originates from the querying peer, and randomly steps forward through the network. Peers that have relevant results send these back directly to the originating peer. Peers participating in the walk occassionally check the satisfaction the originating peer has with respect to the the number of results obtained so far and terminate the walk based on this. [Yang et al. 2006] find that this approach is slow, but multiple walks can be started in parallel to decrease the latency. Similar to random walks [Kalogeraki et al. 2002] propose to forward query messages to a randomly selected fraction of neighbouring peers. However, this still increases messaging costs exponentially when increasing the fraction whereas for random walkers this remains linear.

4.2.8. Reducing Latency and Improving Recall using Directed Walks. [Adamic et al. 2001; Joseph 2002; Kalogeraki et al. 2002; Yang and Garcia-Molina 2002; Tsoumakos and Roussopoulos 2003; Zhong et al. 2003; Zeinalipour-Yazti et al. 2004; Li et al. 2009; Song et al. 2010]

[Adamic et al. 2001] route query messages via high-degree nodes: those with high connectivity, and show that this both decreases search time and increases the network

penetration. [Yang and Garcia-Molina 2002] forward query messages to peers that previously returned the most query results. In a similar vein [Tsoumakos and Roussopoulos 2003] introduce adaptive probabilistic search where each peer maintains a probabilistic routing table for each query that either originated from it or travelled through it. The initial peer that submits a query broadcasts it to all its neighbours, but from there on the query message is forwarded only to the neighbour that has the highest probability of obtaining results based on past feedback. [Zeinalipour-Yazti et al. 2004] build upon this and propose a mechanism where peers actively build profiles of neighbouring peers based on the most recent queries they responded to, similar to [Joseph 2002]. A peer scores incoming queries against the profiles of its neighbours ranking them both qualitatively, based on their cosine similarity, and quantitatively: the number of previously returned results. This outperforms basic flooding, random forwarding [Kalogeraki et al. 2002], and pure quantitative directed routing [Yang and Garcia-Molina 2002].

[Zhong et al. 2003] consider an economic approach to query routing in mobile networks using a simple, cheat-proof, credit-based system which relies on a central credit authority that charges peers for sending messages. Credit can be obtained with real money or by forwarding messages for other peers. [Li et al. 2009] take this work as inspiration and apply it to peer-to-peer networks with local indices. When a peer issues a query it offers a reward for the results. Specifically, neighbouring peers are promised this premium as payment when relevant search results are returned via them. Peers may choose to which other peers they forward a query and do so in return for a part of the premium offered to them. Finally, when the query routing process discovers a peer that has relevant search results, it passes these back along the path to the peer that initiated the query. Along the way each peer is given the promised reward. This reward currency can be used to issue new queries by each peer and thus encourages participation in routing. The authors show that their approach better utilizes the peer-to-peer network's capacity than both query flooding and random walks.

5. EXISTING SYSTEMS

Many peer-to-peer information retrieval systems have been developed for various applications. These systems often borrow elements from file sharing networks and federated information retrieval with various levels of success. Most research systems focus on either the domain of computing grids, digital libraries or the Web.

Table IV lists references to some of the literature that describes several major research systems developed. Figure 3 shows a classification breakdown of each of these peer-to-peer information retrieval systems plus all other research systems discussed.

5.1. Scientific Systems

Although many research systems exist, we restrict ourselves to a subset of them in this section. We discuss systems that stand out because of either their pioneering nature or by their use of an interesting mix of techniques.

Sixearch. One of the first peer-to-peer information retrieval systems was the Infrasearch project, which later became JXTASearch [Waterhouse et al. 2002; Klampanos and Jose 2004] which is also the basis for Sixearch [Akavipat et al. 2006; Lele et al. 2009]. Sixearch consists of several components: a topical crawler, a document indexing system, a retrieval engine, JXTA for peer-to-peer network communication, and a contextual learning system. They use an XML-based architecture and assume that the query consists of a structured customisable set of fields. A book collection could for example have the fields: title, author, et cetera. This approach is not geared too well towards full-text retrieval since it is based on the structure of queries rather than that of

Table IV: Selection of Literature Regarding Several Peer-to-Peer Information Retrieval Systems

Name	References
DCT / ALVIS	[Luu et al. 2006; Skobeltsyn and Aberer 2006]
	[Skobeltsyn et al. 2007; Skobeltsyn et al. 2009]
DHI / SPINA	[Di Buccio et al. 2009]
DTF	[Fuhr 1999; Nottelmann and Fuhr 2007]
pSearch / eSearch	[Tang et al. 2002; Tang and Dwarkadas 2004]
MINERVA	[Bender et al. 2005b; Chernov et al. 2005; Bender et al. 2006]
	[Michel et al. 2006; Bender et al. 2007]
	[Zimmer et al. 2008]
NeuroGrid	[Joseph 2002]
ODISSEA	[Suel et al. 2003; Zhang and Suel 2005]
PHIRST	[Rosenfeld et al. 2009]
PlanetP	[Cuenca-Acuna et al. 2003]
SETS	[Bawa et al. 2003]
Sixearch	[Akavipat et al. 2006; Menczer et al. 2008; Lele et al. 2009]

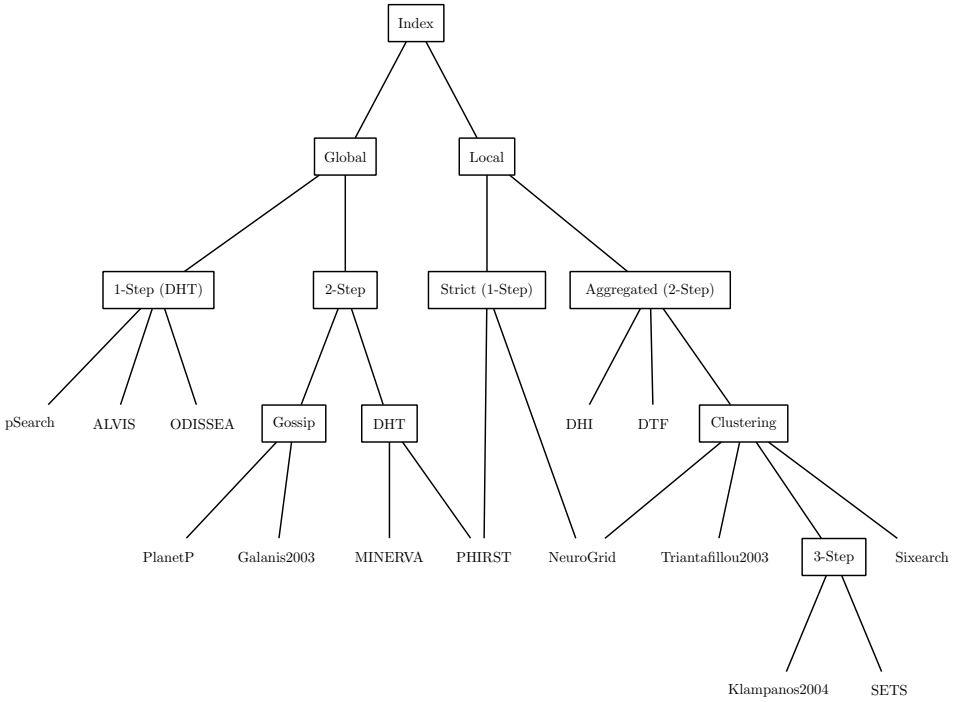


Fig. 3: Classification of peer-to-peer information retrieval research systems. See Section 2.5 and Section 4.2 for an explanation of the rectangular distinctions. Clustering in this diagram means explicit interest-based or content-based clustering and not the random clusters that can occur naturally when using aggregated local indices.

the content shared [Klampanos and Jose 2004]. Supplier peers publish, for each query field, a set of keywords for which they believe they can provide relevant results: their resource description. Consumer peers pose structured queries which are then routed to appropriate supplier peers using hubs. The query routing bases itself on content profiles of neighbouring peers continually improved using reinforcement learning based on past interactions. Fusion of search results returned by multiple peers uses a simple voting algorithm. The authors want to improve their system by focusing on contextual learning and social collaboration. They intend to extend their system with a reputation system as a security component to distinguish spammers from honest peers [Menczer et al. 2008].

ODISSEA. [Suel et al. 2003] introduce the ODISSEA peer-to-peer architecture. Their system consists of two tiers. The lower tier consists of peers that maintain a distributed global index. The postings for a term are always located at a single peer. The upper tier consists of update peers that insert or update documents in the system, like a crawler or a Web server, and query peers that use the lower tier to answer queries. The novelty in their approach is both in the specialisation of peers as well as in their usage of a distributed global term-document index. The specialisations make that in their system the peers responsible for storing, constructing and querying the index are in fact disjunct. This resembles the completely centralised approach commonly used by modern search engines where some machines just store documents in the index, some crawl to keep the index fresh and (external) others only query. In contrast to those systems, ODISSEA offers an open indexing and searching infrastructure in which every machine that speaks the protocol can participate as peer.

When handling multiterm queries, the posting list intersections are conducted in ascending order of term posting list size: from small to large, as this greatly reduces the amount of data that needs to be transferred. Furthermore, they apply top k query processing to minimise bandwidth usage. The authors suggest optimisation of query execution, compression, and pruning techniques as important future work. Furthermore, they state that Web-scale information retrieval is a much more challenging application than file sharing.

MINERVA. [Bender et al. 2005b] assume that each peer performs its own crawls and builds a local index. A peer first searches its own local index to find relevant search results. If these results are unsatisfactory, the peer can consult a global distributed index which contains for each term in the network a list of peers that have relevant documents: a two-step index. This global index also contains statistics regarding the local indices maintained by each peer. The authors show that properly estimating the overlap between search results can reduce the number of peers that need to be contacted for complete recall by more than 60%. However, the lookups in a distributed hash table remain expensive. [Bender et al. 2006] propose to use correlations among individual terms in a query to reduce the number of lookups: the peer that handles the first term in the query also, adaptively, stores what peers to contact for the remaining query terms. This significantly reduces the number of involved peers, while sustaining the same level of recall. Nevertheless, popular terms can cause severe load imbalances if a single peer bears responsibility for storing all postings for one term. [Michel et al. 2005b] propose creating one-step term-document indices in MINERVA for popular terms to reduce response times. Since posting lists are usually scanned sequentially, from the best to worst scoring document for a particular term, they use an order-preserving hash function[2] to store the postings for a term sorted by descending score over multiple peers. The authors apply top k query processing to further reduce

[2]Such a function guarantees that if $a > b$, then $hash\,(a) > hash\,(b)$.

load. This can be further optimised by applying search result caching [Zimmer et al. 2008]: storing cached search results for each complete query on peers that store the postings for one of the query terms. These results contain meta information that helps in judging whether they are still fresh enough and whom to contact for refreshed results. The authors show that cycling out the least frequently used item is the best cache management strategy for a bounded cache. They experiment with both exact caching: matching a multiterm query exactly, and approximate caching: matching term subsets of a multiterm query. They find that both approaches save valuable network resources without compromising result quality.

In later work [Bender et al. 2005a] consider the *novelty* of additional resulting documents in addition to the quality, using a modified federated information retrieval collection selection algorithm. This also appears in [Michel et al. 2006] who focus on further optimizing query routing. Furthermore, they experiment with Bloom filters and Min-Wise Independent Permutations, showing that the latter is better suited for obtaining result set size estimations.

ALVIS. [Luu et al. 2006] introduce the ALVIS Peers system. This is a distributed global index approach, with several innovations. During final result fusion each peer that generated an index entry is contacted and asked to recompute the document score based on global and local statistics, thereby generating globally comparable scores. Instead of storing postings for individual terms, the authors use highly discriminative keys. This introduces the problem of having to store many more keys than in a conventional term-peer index. To mitigate this, in later work [Skobeltsyn et al. 2007; Skobeltsyn et al. 2009] they combine their approach with query-driven indexing storing only popular keys in the index and apply top k result storing. While this has a penalty for less popular, long-tail, queries, [Shokouhi et al. 2007] already showed that query logs can be effectively used to prune irrelevant keys from an index without much performance loss.

PHIRST. The differences between global and local indices give rise to a difficult tradeoff. We have to choose between fast, but costly and inflexible exact search or slow, but inexpensive and flexible approximate search. [Rosenfeld et al. 2009] present an approach to peer-to-peer full-text search that combines global and local indices for different types of terms. They keep only the low- frequency terms in a hash table, while estimating the counts for common terms via limited query flooding. Newly added documents likely contain more well-known highly frequent terms and less new low-frequency terms. Because of this effect they claim that their approach leads to a proportionally smaller index as the number of indexed documents and peers increases compared to a full index kept in a distributed hash table. [Loo et al. 2004; Huebsch et al. 2005] already showed that this hybrid approach improves recall and response times and incurs less bandwidth overhead for search in file sharing.

Klampanos2004. [Klampanos and Jose 2004] attempt to apply standard information retrieval approaches in a peer-to-peer environment. They combine aggregated local indices with content clustering. They assume that each peer indexes its own documents and finds content clusters in its own collection. At the network level each peer joins one or more content-aware groups based on its local clusters. The content-aware groups are potentially overlapping clusters of peers. Each super peer stores the descriptors of each of these groups in the network and given a query can score it against them. A simplified version of Dempster-Shafer theory, a way to combine evidence from multiple sources into one common belief, is used to fuse the results given by multiple peers. This seems to perform well, while contacting a low number of peers: usually one or two, sometimes three and rarely six.

NeuroGrid. [Joseph 2002] introduces an adaptive decentralised search system called NeuroGrid which uses hybrid indexing: initially all the peers have their own local document index, but when they join the network they create a peer index of their neighbouring peers. This closely resembles aggregated local indices, but with each peer functioning as a super peer. Initially a NeuroGrid network is a simple message flooding network. The novelty in the approach is in the adaptive routing of queries. User responses to search results, the absence of positive feedback or explicit negative feedback, are recorded. When NeuroGrid has to select a subset of peers to forward a query to it tries to maximise the chance of receiving positive feedback for the returned results based on these previous experiences. In case of positive feedback the querying peer establishes a direct link to the responding peer in the overlay network. This type of clustering gradually increases connectivity and makes all peers become more knowledgeable concerning the content of their neighbours. This approach also reduces the length of the path that queries need to travel over time. The system prefers reliable peers: those that respond to queries and supply on-topic results of interest to the user. Well-connected peers are more influential on the statistical learning process.

Galanis2003. [Galanis et al. 2003] propose organising all the data sources on the Internet in a large peer-to-peer network where queries are answered by relevant sites. They assume that each peer is essentially an XML search engine that maintains a local index. When a peer joins the network it sends other peers a summary of its data: a small set of selected tags and keywords representative for its content. A join thus generates a wave of messages, making their approach geared towards networks with very low churn. Alternatively such information can also be piggybacked when sending queries as in [Di Buccio et al. 2009]. Peers also acquire, initially from neighbouring peers, content summaries of other peers in the system and maintain their own peer index. The authors experiment with replicating summaries to every peer and to peer subsets of various sizes. Their results suggest that using replication to every peer outperforms that of using subsets, although using large subsets can approach this performance. They compare full replication aggregated indices with a strict local indices approach and show aggregation increases query throughput with 2071% and offers 72 times faster response times.

Triantafillou2003. [Triantafillou et al. 2003] focus on enforcing fair load distribution among peers. They cluster documents into semantic categories and cluster peers based on the categories of documents they offer. The authors emphasise the need to impose some logical system structure to achieve high performance in a peer-to-peer information retrieval network. Peers maintain a document index, that maps document identifiers to categories, a cluster index that maps categories to cluster identifiers, and a peer index that maps cluster identifiers to peers. The terms in a query are first mapped to categories, then to clusters and finally to a random peer within the relevant clusters. This random peer tries to satisfy the query with its own local results, but if too few are available it forwards the query to neighbouring nodes in the same cluster. This repeats until there are sufficient results. Since selection is random each peer in a cluster is equally likely to be picked which achieves load balancing among peers within the same cluster. Peers are assigned to clusters based on the categories of documents they share. For load balancing among clusters the authors introduce the *fairness index* and a greedy algorithm to maximise this called MaxFair which also compensates for peers with different processing power, content distribution and storage capacities. The most powerful peer in each cluster is designated as leader which participates in the MaxFair algorithm. Categories may be dynamically reassigned to a different cluster to improve fairness based on the load of each cluster. They show that their approach is capable of maintaining fairness even when peers and document collections change.

5.2. Non-Scientific Systems

Various developed systems exist that do not have direct scientific roots. In this section we list several of the better known systems. Although we attempt to give some details about the underlying technology used, it is often a bit harder to classify these systems as operational details are sometimes missing or not well documented.

YaCy. `www.yacy.net`

In YaCy each peer crawls parts of the Web and builds a local index. When connected to the larger YaCy network the local index entries are injected into a distributed global index with a high level of redundancy to ensure continuous availability. YaCy uses no centralised servers, but relies on a select set of machines that maintain seed lists for bootstrapping peers into the network. To protect user privacy it does not index deep Web pages by default. However, parameters can be changed. YaCy is an open project with transparent protocols and positions itself as a counter-movement against the increasing influence of, and dependency on, proprietary search engines. As of July 2011 it consists of about 600 peers, which indexed 1.4 billion documents and serve about 130 000 queries daily.

Seeks. `www.seeks-project.info`

This aims to design and develop an open peer-to-peer network that provides social search overlay: clustering users with similar queries so they can share both query results, similar to interest-based clustering, but also their experiences with those results: making it a social network. It aims to make real-time, decentralised, Web search a reality. To protect the privacy of users the queries are hashed. Seeks performs no crawling, instead relying solely on users to push content into the network. Although it is initially populated with search results from major search engines. Seeks uses a distributed global index and is usable and under active development as of 2012.

Faroo. `www.faroo.com`

This is a proprietary peer-to-peer search engine that uses a distributed global index and aims to 'democratise search'. They perform distributed crawling and ranking. Faroo encrypts queries and results for privacy protection. They claim to be the largest peer-to-peer search engine with as many as 2 million peers.

6. THE FUTURE OF PEER-TO-PEER INFORMATION RETRIEVAL

6.1. Challenges

We have already seen some challenges that apply to peer-to-peer networks in general. In this section we discuss a subset of these aspects more important to peer-to-peer information retrieval.

Latency. In peer-to-peer information retrieval the latency that occurs when executing searches is dominated by the number of peers involved in routing and processing queries. We have seen that local and global indices are suitable for different types of queries, and that there are many optimisations that can be applied to reduce the cost of storing and transmitting index information. Nevertheless, the challenge of optimally combining these techniques, and finding new ones, to keep latency within acceptable bounds remains. The reason for this is primarily that there is no one good solution for all cases and that the amount of information to index is increasing leading to greater latency problems. For any search system, it is important to serve search results quickly *without* compromising too much on the quality of the results. The technical causes of delay are irrelevant to users. After entering a query, the results should appear in at most 2 seconds. However, ideally the results are perceived as appearing instantaneously to compete with centralised solutions, which means a delay of 0.1 sec-

onds. Anything below that is unlikely to positively impact the user experience [Nah 2004]. Most existing solutions rightly focus on reducing the number of hops and/or using parallelisation to reduce latency. Efficient query routing is a challenge specific to peer-to-peer information retrieval and directly tied to latency. More research in temporal aspects of querying could lead to more optimal tailored solutions.

Freshness. Keeping the index fresh is a challenge for every search engine which is commonly the responsibility of the engine's Web crawling component. The index needs to be representative of the indexed websites, without incurring too much load on those sites to detect changes. Some Web documents change quickly and some change rarely, and not every change that occurs is significant enough to warrant an index update [Risvik and Michelsen 2002]. In the ideal situation websites participate cooperatively in a peer-to-peer network and signal significant changes to themselves to the network. This would remove the need for crawling. However, peer-to-peer Web search engines will initially have to cope with the existing situation. Having peers perform their own crawl seems realistic, but introduces the same problems as conventional Web crawling. Since many updates can occur due to changing documents it is important that the index used has minimal mutation overhead. Separate indexing strategies could be used for fast and slow changing Web documents. A further challenge is caching of postings lists or search results. These mechanisms decrease latency, but do so at the expense of freshness.

Evaluation. Even though a simulation can fix many of the free variables of a peer-to-peer network, for rigorous comparison the same data needs to be used. There is a need for a common collection, a way to distribute this collection over multiple peers and a query set. There have been at least two attempts at establishing such a benchmark [Klampanos et al. 2005; Neumann et al. 2006], although they have not yet seen widespread adoption. [Klampanos et al. 2005] state that evaluating peer-to-peer information retrieval systems is a demanding and neglected task. They establish a number of different document test beds for evaluating these systems. They state that evaluation of these networks is particularly hard for several reasons. Firstly, they are assumed to be very large which makes simulation more difficult. Secondly, they are subject to churn caused by normal peer on-off cycles and peers that crash or malfunction. Unfortunately, the impact of churn is not well investigated in peer-to-peer information retrieval experiments, as most assume an always-on environment [Zeinalipour-Yazti et al. 2004]. Thirdly, documents are not likely to be randomly placed at peers, instead their distribution is influenced by previous location and retrieval, and replication. Lastly, simulating user behaviour is complex, for example: realistically simulating how both collections and query frequencies change over time. This is usually circumvented by reflecting behaviour in the document distribution. However, it is difficult to reflect the application scenario such that the results can be conclusive.

Different types of peer-to-peer information retrieval networks have different document distributions. [Klampanos et al. 2005] present standardised distributions for three of these derived from the WT10g collection [Bailey et al. 2003] using about 1.7 million documents. Firstly, the Web domain where the distribution of documents follows a power-law. Secondly, loosely controlled grid networks with a uniform distribution of documents that impose an equal load on each peer. Thirdly, digital libraries where the distribution also follows a power law, although less extreme than for the Web domain. Additionally, digital libraries have fewer peers which each share a significantly larger amount of documents compared to the other cases. Replication is simulated in all cases by exploiting interdomain links. The Web and grid scenarios use about 11,680 peers, whereas 1,500 are used for the digital library case. [Lu 2007] also presents a test bed for digital libraries with 2,500 peers.

6.2. Key Focus Areas

We believe there are several key areas which are important to focus on today to be able to create the peer-to-peer information retrieval systems of tomorrow. The following list is based on existing research and our own insights:

— Combining the strengths of *global and local indices* and developing algorithms to easily shift appropriate content from one to the other based on term or query popularity. Existing systems do not scale well because they are solely based on either flooding the network with queries or because they require some form of global knowledge [Zeinalipour-Yazti et al. 2004].
— No architecture exists that offers the best solution for all peer-to-peer information retrieval problems, different architectures apply to different situations.
— Although good scalability properties are inherent to the peer-to-peer paradigm, systems that wish to support Web-scale search need to focus on effectively distributing their load over a high number of peers: hundreds of thousands to millions [Triantafillou et al. 2003]. An important reason for this is that peers are heterogeneous in terms of capacity and connectivity and are not dedicated server machines: they have to perform many other tasks as well.
— Focusing on *search results* instead of documents. This means shifting attention to networks that provide access to external documents emphasizing the *search* task: the core of peer-to-peer information retrieval.
— Investigating and improving the performance of *search result caches*. It is important to achieve a good balance between providing results which are sufficiently fresh, and not taxing the network for updating those results. This also should depend on the, predicted, mutation frequency of the resources pointed to.
— Improving handling of peer *heterogeneity* in Web search. A few peers have a lot of documents, whereas many peers have much smaller collections. These smaller collections are often specialised, making them appropriate for more specific queries.
— Applying *clustering* at various levels as it simplifies both the construction of resource descriptions and query routing, resulting in reduced latency.
— Improving both *topology* and *query routing*, particularly for avoiding and routing around hotspots in networks. A good topology favours both effectiveness and efficiency, by making it possible for a query to reach a relevant target peer in few steps.
— Focusing on *precision over recall*. Achieving a hundred percent recall in peer-to-peer systems would involve searching in all indices and is far too costly [Klampanos et al. 2005]. It is also unnecessary if the quality of the returned results is high enough. Although, this requires better result fusion techniques. One should realise that Web search users do not browse beyond the first page, but reformulate their queries.
— Developing *real-time distributed relevance-feedback mechanisms* [Klampanos and Jose 2007]. Ideally, search result quality continually improves based on user feedback as is common for centralised search engines. The emerging trend of coupling this to social networks could be further explored.
— Creating a number of large standardised *test collections* which apply to different types of peer-to-peer information retrieval networks. The work of [Klampanos et al. 2005] provides a good start, but is still somewhat conservative with respect to scale.
— Focusing on *tangible benefits* of peer-to-peer networks rather than ethics or 'coolness', giving users a proper incentive to search using such networks over other solutions. This is particularly important for peer-to-peer Web search engines.
— Realising that any Web search service is a form of *adversarial information retrieval*: companies and people, suppliers of information, have an incentive to appear high up in rankings [Craswell and Hawking 2009]. Use this fact to improve the quality of service for the users.

7. CONCLUSION

In the peer-to-peer paradigm each node is equal and can both supply and consume resources. A peer-to-peer network typically consists of thousands of low-cost machines all with different processing and storage capacities as well as different link speeds. The advantages of these networks are that they have no central point of failure, are self-organizing and self-scaling. In this article we have focused on information retrieval networks used to exchange public data. We have identified a number of key challenges for peer-to-peer networks with respect to usage guarantees, behaviour of peers, and evaluation. We presented the three main tasks that every peer-to-peer system performs: searching, locating and transferring. Furthermore, we organised the main architectures around index placement: the centralised global index, the distributed global index, strict local indices and aggregated local indices. Each has different consequences for query routing and processing. We have also shown the difference between a one-step index, where keys map directly to documents, and a two-step index, where the keys map to peers and the peers themselves contain document mappings.

In peer-to-peer information retrieval the central task is searching, in contrast with file sharing systems where the emphasis is on transferring. Many of the challenges in federated information retrieval significantly overlap with those in peer-to-peer information retrieval: resource description, collection selection and result merging. However, in federated information retrieval the mediator party plays an important role, and additionally exhibits a strict division between the consumers and providers of information.

In existing peer-to-peer information retrieval research we find both partition-by-document and partition-by-keyword indexing approaches. We have shown various optimisations that can be applied to reduce bandwidth and latency and to improve the quality and quantity of the search results returned. Besides this we have given an overview and classification of existing systems. Finally, we have discussed the future of peer-to-peer information retrieval indicating specific challenges and key areas to focus on. The most important of these are: emphasising precision over recall, focusing on search results instead of documents, combining the strengths of local and global indices, applying clustering and using relevance feedback.

While peer-to-peer technology has seen widespread adoption in file sharing, it is not often used for solving information retrieval problems. This is unfortunate as it has the potential to offer a robust solution to the ethical and technical problems that plague centralised solutions and thus deserves more attention. We believe one of the primary reasons for its unpopularity in science is because the field lacks a clear definition that distinguishes it from related and overlapping fields like general peer-to-peer systems and federated information retrieval. Moreover, there was no overview of what has been done so far and what choices can be made when implementing a practical system. Both of these aspects are important for future research and development of real-world systems. This article provides a solid basis for this.

References

ABERER, K. AND HAUSWIRTH, M. 2002. An overview on peer-to-peer information systems. In *Proceedings of WDAS*.

ADAMIC, L. A., LUKOSE, R. M., PUNIYANI, A. R., AND HUBERMAN, B. A. 2001. Search in power-law networks. *Physical Review E 64,* 4, 046135–1–046135–8.

AKAVIPAT, R., WU, L.-S., MENCZER, F., AND MAGUITMAN, A. G. 2006. Emerging semantic communities in peer web search. In *Proceedings of P2PIR*. ACM, New York, NY, USA, 1–8.

BAILEY, P., CRASWELL, N., AND HAWKING, D. 2003. Engineering a multi-purpose test collection for web retrieval experiments. *Information Processing & Management 39,* 6, 853–871.

BALKE, W.-T., NEJDL, W., SIBERSKI, W., AND THADEN, U. 2005. Progressive distributed top-k retrieval in peer-to-peer networks. In *Proceedings of ICDE*. IEEE Computer Society, Washington, DC, USA, 174–185.

BAWA, M., MANKU, G. S., AND RAGHAVAN, P. 2003. Sets: Search enhanced by topic segmentation. In *Proceedings of SIGIR*. ACM, New York, NY, US, 306–313.

BENDER, M., MICHEL, S., TRIANTAFILLOU, P., AND WEIKUM, G. 2007. Design alternatives for large-scale web search: Alexander was great, aeneas a pioneer, and anakin has the force. In *Proceedings of LSDIR2007 Workshop*.

BENDER, M., MICHEL, S., TRIANTAFILLOU, P., WEIKUM, G., AND ZIMMER, C. 2005a. Improving collection selection with overlap awareness in p2p search engines. In *Proceedings of SIGIR*. ACM, New York, NY, US, 67–74.

BENDER, M., MICHEL, S., TRIANTAFILLOU, P., WEIKUM, G., AND ZIMMER, C. 2005b. Minerva: Collaborative p2p search. In *Proceedings of VLDB (Demos)*. 1263–1266.

BENDER, M., MICHEL, S., TRIANTAFILLOU, P., WEIKUM, G., AND ZIMMER, C. 2006. P2p content search: Give the web back to the people. In *Proceedings of IPTPS*.

BLOOM, B. H. 1970. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM 13,* 7, 422–426.

CALLAN, J. 2000. *Distributed Information Retrieval*. Advances in Information Retrieval. Kluwer Academic Publishers, Chapter 5.

CALLAN, J., LU, Z., AND CROFT, W. B. 1995. Searching distributed collections with inference networks. In *Proceedings of SIGIR*. ACM Press, 21–28.

CHERNOV, S., SERDYUKOV, P., BENDER, M., MICHEL, S., WEIKUM, G., AND ZIMMER, C. 2005. Database selection and result merging in p2p web search. In *Proceedings of DBISP2P 2005*. Springer Verlag, Heidelberg, DE, 26–37.

CLARKE, I., SANDBERG, O., WILEY, B., AND HONG, T. 2001. Freenet: A distributed anonymous information storage and retrieval system. In *Proceedings of PET*. 46–66.

COHEN, B. 2003. Incentives build robustness in bittorrent. In *Proceedings of P2PEcon*.

CRASWELL, N. AND HAWKING, D. 2009. Web information retrieval. In *Information Retrieval: Searching in the 21st Century*. Wiley, UK, 85–101.

CRESPO, A. AND GARCIA-MOLINA, H. 2004. Semantic overlay networks for p2p systems. In *Proceedings of AP2PC*. Springer, Heidelberg, DE, 1–13.

CUENCA-ACUNA, F. M., MARTIN, R. P., AND NGUYEN, T. D. 2003. Planetp: Using gossiping to build content addressable peer-to-peer information sharing communities. In *Proceedings of HPDC*.

DASWANI, N., GARCIA-MOLINA, H., AND YANG, B. 2003. Open problems in data-sharing peer-to-peer systems. In *Proceedings of ICDT*. Springer, Heidelberg, DE, 1–15.

DI BUCCIO, E., MASIERO, I., AND MELUCCI, M. 2009. Improving information retrieval effectiveness in peer-to-peer networks through query piggybacking. In *Proceedings of ECDL*. 420–424.

DU, A. AND CALLAN, J. 1998. Probing a collection to discover its language model. Tech. Rep. UM-CS-1998-029, University of Massachusetts, Amherst, MA, US.

FAGIN, R., LOTEM, A., AND NAOR, M. 2001. Optimal aggregation algorithms for middleware. In *Proceedings of PODS*. ACM, New York, NY, US, 102–113.

FUHR, N. 1999. A decision-theoretic approach to database selection in networked ir. *ACM Transactions on Information Systems 17,* 3, 229–249.

GALANIS, L., WANG, Y., JEFFERY, S., AND DEWITT, D. 2003. Processing queries in a large peer-to-peer system. In *Proceedings of CAiSE*. Springer, Heidelberg, DE, 273–288.

GIRDZIJAUSKAS, S., GALUBA, W., DARLAGIANNIS, V., DATTA, A., AND ABERER, K. 2011. Fuzzynet: Ringless routing in a ring-like structured overlay. *Peer-to-Peer Networking and Applications 4,* 3, 259–273.

GRAVANO, L., CHANG, K. C.-C., GARCIA-MOLINA, H., LAGOZE, C., AND PAEPCKE, A. 1997. Stanford protocol proposal for internet retrieval and search. Tech. rep., Stanford University.

GRAVANO, L., GARCIA-MOLINA, H., AND TOMASIC, A. 1999. Gloss: Text-source discovery over the internet. *ACM Transactions on Database Systems 24,* 2, 229–264.

HUEBSCH, R., CHUN, B., HELLERSTEIN, J. M., LOO, B. T., MANIATIS, P., ROSCOE, T., SHENKER, S., STOICA, I., AND YUMEREFENDI, A. R. 2005. The architecture of pier: an internet-scale query processor. In *Proceedings of CIDR*.

JOSEPH, S. 2002. Neurogrid: Semantically routing queries in peer-to-peer networks. In *Proceedings of Networking Workshops*. 202–214.

KALOGERAKI, V., GUNOPULOS, D., AND ZEINALIPOUR-YAZTI, D. 2002. A local search mechanism for peer-to-peer networks. In *Proceedings of CIKM*. ACM, 300–307.

KAMVAR, S. D., SCHLOSSER, M. T., AND GARCIA-MOLINA, H. 2003. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of WWW*.

KIRSCH, S. T. 1997. Document retrieval over networks wherein ranking and relevance scores are computed at the client for multiple database documents.

KLAMPANOS, I. AND JOSE, J. M. 2007. An evaluation of a cluster-based architecture for peer-to-peer information retrieval. In *Proceedings of DEXA*. 380–391.

KLAMPANOS, I. A. AND JOSE, J. M. 2004. An architecture for information retrieval over semi-collaborating peer-to-peer networks. In *Proceedings of SAC*. ACM, New York, NY, US, 1078–1083.

KLAMPANOS, I. A., POZNAŃSKI, V., JOSE, J. M., AND DICKMAN, P. 2005. A suite of testbeds for the realistic evaluation of peer-to-peer ir systems. In *Proceedings of ECIR*. 38–51.

KLEINBERG, J. M. 2006. Complex networks and decentralized search algorithms. In *Proceedings of ICM*.

KRISHNAN, R., SMITH, M. D., TANG, Z., AND TELANG, R. 2002. The virtual commons: Why free-riding can be tolerated in file sharing networks. In *Proceedings of ICIS*.

KUROSE, J. F. AND ROSS, K. W. 2003. *Computer Networking: A Top-Down Approach Featuring the Internet* 2nd Ed. Addison-Wesley.

LELE, N., WU, L.-S., AKAVIPAT, R., AND MENCZER, F. 2009. Sixearch.org 2.0 peer application for collaborative web search. In *Proceedings of HT*. ACM, New York, NY, US, 333–334.

LI, C., YU, B., AND SYCARA, K. 2009. An incentive mechanism for message relaying in unstructured peer-to-peer systems. *Electronic Commerce Research and Applications 8,* 6, 315 – 326.

LI, J., LOO, B. T., JOSEPH, L., HELLERSTEIN, J. M., KARGER, D. R., MORRIS, R., AND KAASHOEK, M. F. 2003. On the feasibility of peer-to-peer web indexing and search. In *Proceedings of IPTPS*. Lecture Notes in Computer Science Series, vol. 2735. Springer, 207–215.

LOO, B. T., HUEBSCH, R., STOICA, I., AND HELLERSTEIN, J. M. 2004. The case for a hybrid p2p search infrastructure. In *Proceedings of IPTPS*. 141–150.

LU, J. 2007. Full-text federated search in peer-to-peer networks. Ph.D. thesis, Carnegie Mellon University.

LU, J. AND CALLAN, J. 2006. Full-text federated search of text-based digital libraries in peer-to-peer networks. *Information Retrieval 9,* 4, 477–498.

LU, J. AND CALLAN, J. 2007. Content-based peer-to-peer network overlay for full-text federated search. In *In Proceedings of RIAO*.

LUA, E. K., CROWCROFT, J., PIAS, M., SHARMA, R., AND LIM, S. 2005. A survey and comparison of peer-to-peer overlay network schemes. *Communications Surveys & Tutorials 7,* 2, 72–93.

LUU, T., KLEMM, F., PODNAR, I., RAJMAN, M., AND ABERER, K. 2006. Alvis peers: A scalable full-text peer-to-peer retrieval engine. In *Proceedings of P2PIR*. ACM, New York, NY, US, 41–48.

LV, Q., CAO, P., COHEN, E., LI, K., AND SHENKER, S. 2002. Search and replication in unstructured peer-to-peer networks. In *Proceedings of ICS*. ACM, New York, NY, US, 84–95.

MENCZER, F., WU, L.-S., AND AKAVIPAT, R. 2008. Intelligent peer networks for collaborative web search. *Artificial Intelligence Magazine 29,* 3, 35–46.

MICHEL, S., BENDER, M., TRIANTAFILLOU, P., AND WEIKUM, G. 2006. Iqn routing: Integrating quality and novelty in p2p querying and ranking. In *Proceedings of EDBT*. Lecture Notes in Computer Science Series, vol. 3896. Springer, 149–166.

MICHEL, S., TRIANTAFILLOU, P., AND WEIKUM, G. 2005a. Klee: A framework for distributed top-k query algorithms. In *Proceedings of VLDB*. 637–648.

MICHEL, S., TRIANTAFILLOU, P., AND WEIKUM, G. 2005b. Minerva infinity: A scalable efficient peer-to-peer search engine. In *Proceedings of Middleware 2005*. Springer, Heidelberg, DE, 60–81.

MONNERAT, L. AND AMORIM, C. 2009. Peer-to-peer single hop distributed hash tables. In *Proceedings of GLOBECOM 2009*. 1–8.

NAH, F. F.-H. 2004. A study on tolerable waiting time: How long are web users willing to wait? *Behaviour & Information Technology 23,* 3, 153–163.

NAICKEN, S., BASU, A., LIVINGSTON, B., AND RODHETBHAI, S. 2006. A survey of peer-to-peer network simulators. In *Proceedings of PGNet*. EPSRC.

NAICKEN, S., LIVINGSTON, B., BASU, A., RODHETBHAI, S., WAKEMAN, I., AND CHALMERS, D. 2007. The state of peer-to-peer simulators and simulations. *SIGCOMM Computer Communication Review 37,* 2, 95–98.

NEUMANN, T., BENDER, M., MICHEL, S., WEIKUM, G., BONNET, P., AND MANOLESCU, I. 2006. A reproducible benchmark for p2p retrieval. In *Proceedings of EXPDB*. ACM.

NOTTELMANN, H. AND FUHR, N. 2007. A decision-theoretic model for decentralised query routing in hierarchical peer-to-peer networks. In *Proceedings of ECIR*. 148–159.

OULASVIRTA, A., HUKKINEN, J. P., AND SCHWARTZ, B. 2009. When more is less: The paradox of choice in search engine use. In *Proceedings of SIGIR*. ACM, New York, NY, US, 516–523.

REYNOLDS, P. AND VAHDAT, A. 2003. Efficient peer-to-peer keyword searching. In *Proceedings of Middleware*. Lecture Notes in Computer Science Series, vol. 2672. Springer, 997–997.

RISSON, J. AND MOORS, T. 2006. Survey of research towards robust peer-to-peer networks: Search methods. *Computer Networks 50,* 17, 3485–3521.

RISVIK, K. M. AND MICHELSEN, R. 2002. Search engines and web dynamics. *Computer Networks 39,* 3, 289–302.

ROSENFELD, A., GOLDMAN, C. V., KAMINKA, G. A., AND KRAUS, S. 2009. Phirst: A distributed architecture for p2p information retrieval. *Information Systems 34,* 2, 290–303.

SHOKOUHI, M. AND ZOBEL, J. 2007. Federated text retrieval from uncooperative overlapped collections. In *Proceedings of SIGIR*. ACM, New York, NY, US, 495–502.

SHOKOUHI, M., ZOBEL, J., TAHAGHOGHI, S. M. M., AND SCHOLER, F. 2007. Using query logs to establish vocabularies in distributed information retrieval. *Information Processing & Management 43,* 1, 169–180.

SI, L. AND CALLAN, J. 2003a. Relevant document distribution estimation method for resource selection. In *Proceedings of SIGIR*. ACM, New York, NY, USA, 298–305.

SI, L. AND CALLAN, J. 2003b. A semisupervised learning method to merge search engine results. *ACM Transactions on Information Systems 21,* 4, 457–491.

SKOBELTSYN, G. AND ABERER, K. 2006. Distributed cache table: efficient query-driven processing of multi-term queries in p2p networks. In *Proceedings of P2PIR*. 33–40.

SKOBELTSYN, G., LUU, T., ABERER, K., RAJMAN, M., AND PODNAR ŽARKO, I. 2007. Query-driven indexing for peer-to-peer text retrieval. In *Proceedings of WWW*. ACM, New York, NY, US, 1185–1186.

SKOBELTSYN, G., LUU, T., PODNAR ŽARKO, I., RAJMAN, M., AND ABERER, K. 2009. Query-driven indexing for scalable peer-to-peer text retrieval. *Future Generation Computer Systems 25*, 89–99.

SONG, W., ZENG, X., HU, W., CHEN, Y., WANG, C., AND CHENG, F. 2010. Resource search in peer-to-peer network based on power law distribution. In *Proceedings of NSWCTC*. 53–56.

SRIPANIDKULCHAI, K., MAGGS, B. M., AND ZHANG, H. 2003. Efficient content location using interest-based locality in peer-to-peer systems. In *Proceedings of INFOCOM*.

STEINER, M., EN-NAJJARY, T., AND BIERSACK, E. W. 2007. Exploiting kad: possible uses and misuses. *SIGCOMM Computer Communication Review 37*, 65–70.

STOICA, I., MORRIS, R., KARGER, D. R., KAASHOEK, M. F., AND BALAKRISHNAN, H. 2001. Chord: A scalable peer-to-peer lookup service for internet applications. *SIGCOMM Computer Communication Review 31,* 4, 149–160.

STUTZBACH, D. AND REJAIE, R. 2006. Understanding churn in peer-to-peer networks. In *Proceedings of IMC*. 189–202.

SUEL, T., MATHUR, C., WU, J.-W., ZHANG, J., DELIS, A., KHARRAZI, M., LONG, X., AND SHANMUGASUNDARAM, K. 2003. Odissea: A peer-to-peer architecture for scalable web search and information retrieval. In *Proceedings of WebDB*. 67–72.

TANG, C. AND DWARKADAS, S. 2004. Hybrid global-local indexing for efficient peer-to-peer information retrieval. In *Proceedings of NSDI*.

TANG, C., XU, Z., AND MAHALINGAM, M. 2002. psearch: Information retrieval in structured overlays. In *Proceedings of HotNets-I*.

TIGELAAR, A. S. AND HIEMSTRA, D. 2010. Query-based sampling using snippets. In *Proceedings of LSD-SIR*.

TIGELAAR, A. S. AND HIEMSTRA, D. 2011. Query load balancing by caching search results in peer-to-peer information retrieval networks. In *Proceedings of DIR*. 28–31.

TIGELAAR, A. S., HIEMSTRA, D., AND TRIESCHNIGG, D. 2011. Search result caching in peer-to-peer information retrieval networks. In *Proceedings of IRFC*. Lecture Notes in Computer Science Series, vol. 6653. 134–138.

TIRADO, J. M., HIGUERO, D., ISAILA, F., CARRETERO, J., AND IAMNITCHI, A. 2010. Affinity p2p: A self-organizing content-based locality-aware collaborative peer-to-peer network. *Computer Networks 54,* 12, 2056–2070.

TRIANTAFILLOU, P., XIRUHAKI, C., KOUBARAKIS, M., AND NTARMOS, N. 2003. Towards high performance peer-to-peer content & resource sharing systems. In *Proceedings of CIDR*. 120–132.

TSOUMAKOS, D. AND ROUSSOPOULOS, N. 2003. Adaptive probabilistic search for peer-to-peer networks. In *Proceedings of P2P*. IEEE Computer Society, 102–110.

VAN HEERDE, H. J. W. 2010. Privacy-aware data management by means of data degradation -making private data less sensitive over time. Ph.D. thesis, Univ. of Twente, Enschede.

WATERHOUSE, S., DOOLIN, D. M., KAN, G., AND FAYBISHENKO, Y. 2002. Distributed search in p2p networks. *IEEE Internet Computing 6,* 1, 68–72.

XU, J. AND CROFT, W. B. 1999. Cluster-based language models for distributed retrieval. In *Proceedings of SIGIR*. ACM, 254–261.

YANG, B. AND GARCIA-MOLINA, H. 2002. Efficient search in peer-to-peer networks. *Proceedings of ICDS*.

YANG, Y., DUNLAP, R., REXROAD, M., AND COOPER, B. F. 2006. Performance of full text search in structured and unstructured peer-to-peer systems. In *Proceedings of INFOCOM*. 1–12.

YUWONO, B. AND LEE, D. L. 1997. Server ranking for distributed text retrieval systems on the internet. In *Proceedings of DASFAA*. World Scientific Press, 41–50.

ZEINALIPOUR-YAZTI, D., KALOGERAKI, V., AND GUNOPULOS, D. 2004. Information retrieval techniques for peer-to-peer networks. *Computing in Science and Engineering 6*, 20–26.

ZHANG, J. AND SUEL, T. 2005. Efficient query evaluation on large textual collections in a peer-to-peer environment. In *Proceedings of P2P*. IEEE Computer Society, Washington, DC, US, 225–233.

ZHONG, S., CHEN, J., AND YANG, Y. R. 2003. Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks. In *Proceedings of INFOCOM*.

ZIMMER, C., BEDATHUR, S., AND WEIKUM, G. 2008. Flood little, cache more: Effective result-reuse in p2p ir systems. In *Proceedings of DASFAA*. Springer-Verlag, Heidelberg, DE, 235–250.

Received July 2011; revised -; accepted December 2011