pFusion: A P2P Architecture for Internet-Scale Content-Based Search and Retrieval

Demetrios Zeinalipour-Yazti, *Member, IEEE,* Vana Kalogeraki, *Member, IEEE,* and
Dimitrios Gunopulos, *Member, IEEE*

***Abstract*— The emerging Peer-to-Peer (P2P) model has become a very powerful and attractive paradigm for developing Internet-scale systems for sharing resources, including files and documents. The distributed nature of these systems, where nodes are typically located across different networks and domains, inherently hinders the efficient retrieval of information. In this paper we consider the effects of topologically-aware overlay construction techniques on efficient peer-to-peer keyword search algorithms. We present the *pFusion* (Peer Fusion) architecture that aims to efficiently integrate heterogeneous information that is geographically scattered on peers of different networks. Our approach builds on work in unstructured peer-to-peer systems, and uses only local knowledge. Our empirical results, using the pFusion middleware architecture and datasets from AKAMAI, NLANR and TREC, show that the architecture we propose is both efficient and practical.**

***Index Terms*— Information Retrieval, Peer-to-Peer, Overlay Construction Algorithms.**

I. Introduction

The worldwide infrastructure of computers and networks creates exciting opportunities for collecting vast amounts of data and for sharing computers and resources on an unprecedented scale. In the last few years, the emerging Peer-to-Peer (P2P) model has become a very powerful and attractive paradigm for developing Internet-scale file systems [34], [35], [41] and sharing resources (i.e., CPU cycles, memory, storage space, network bandwidth) over large scale geographical areas. The basic idea is that an overlay network of nodes (peers) is constructed on top of heterogeneous operating systems and networks. Overlays are flexible and deployable approaches that allow users to perform distributed operations without modifying the underlying physical network.

The first wave of P2P systems implemented *unstructured* P2P overlays in which no global structure or knowledge is maintained. To search for data or resources, messages are sent over multiple hops from one peer to another with each peer responding to queries for information it has stored locally. *Structured* P2P overlays [34], [35], [41] implement a distributed hash table data structure in which every data item can be located within a small number of hops, at the expense of keeping some state information locally at the nodes.

Unstructured P2P systems [6], [9], [10], [50], [54] are very effective infrastructures to share and store documents, because their decentralized nature allows easy additions, updates, increased storage, and offers fault-tolerant properties through the use of replication and caching. In addition, recent efforts based on caching [22] and other heuristics [50] have significantly improved the query routing problem in unstructured P2P systems as well. An important problem that such systems have not fully considered, is how the heterogeneity of the underlying infrastructure affects the performance of the information retrieval algorithms implemented on top of these networks. The P2P infrastructure can encompass resources with different processing and communication capabilities, located across different geographical areas. As a result, retrieving documents over such *Internet-scale* environments is subject to greater variations due to unpredictable communication latencies, excessive resource consumption and changing resource availability.

In this paper we focus on techniques for *distributed keyword search*, that is, we aim to find the documents that contain a given set of query terms when the collection of documents is distributed. Formally, assuming that D_u is a set of documents that are stored on peer u, and each document d is characterized by a set of keywords, the result to a query q (itself as a Boolean expression of keywords), should be the *answer set* $\{(d,u)|u$ is a peer and q $\subset$ $s(d)$ and $d \in D_u\}$, where $s(d)$ is the (unordered) set of keywords in d. To motivate our description we consider two popular applications, *Personal Video Sharing* and *Citizen Journalism*, both of which currently support keyword searches over centralized infrastructures. We explain how these services could optimize their operation through the deployment of topologically-aware P2P networks.

Personal Video Sharing: Web sites, such as Youtube.com [51] and Yahoo Video [49], allow users to *upload*, *search*, *browse* and *view on-demand* the video clips of other users through a keyword-based search interface. Such systems typically exploit a centralized storage and retrieval infrastructure which has a number of disadvantages and limitations: i) the service can easily become a bottleneck during periods of high demand and is also a single point of failure; ii) the infrastructure is expensive and requires extensive administration and iii) the content can be censored. On the contrary, we model such a service on the premise of an unstructured P2P system, where each user stores locally its own video clips and performs the *search* and *retrieval* functions through other participating users in a Gnutella-like fashion [15].

An important point in such Internet-scale applications, is that the large-scale data transfer and retrieval can be expensive

D. Zeinalipour-Yazti is with the Department of Computer Science at the University of Cyprus; email: dzeina@cs.ucy.ac.cy; V. Kalogeraki and D. Gunopulos are with the Department of Computer Science & Engineering at the University of California - Riverside; email:{vana,dg}@cs.ucr.edu

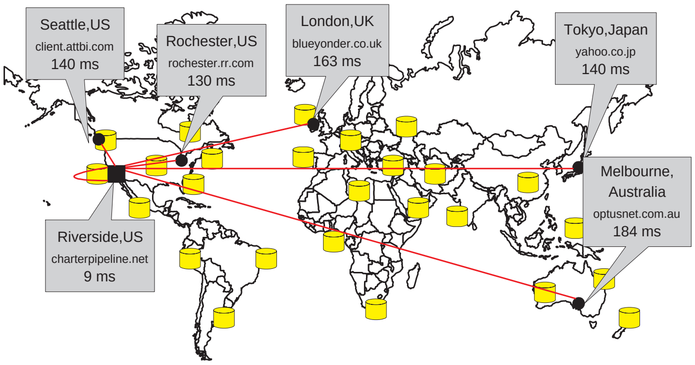

Fig. 1. A peer in Riverside, CA (USA) that is connected to five other peers. In *Internet-scale* applications the large-scale data transfer and retrieval can be expensive when network connections among the *clients* are arbitrary due to unpredictable communication latencies, excessive resource consumption and changing resource availability in inter-domain routing.

when the network connections among the *clients* are arbitrary, due to unpredictable communication latencies, excessive resource consumption and changing resource availability in inter-domain routing. Thus, we seek to optimize the overlay by establishing connections between peers based on the criterion of *network proximity*. In particular, peers minimize the network distance from their neighboring nodes by establishing connections to nodes that belong to the same domain. For example, a node in the Rochester NY (USA) subdomain of the RoadRunner ISP (`rochester.rr.com`), tries to establish overlay connections with other nodes in this same domain. In Section VII, we will show that such a construction provides tremendous reductions in resource consumption and offers an improved user experience. Note that both of these characteristics are essential in a video sharing context where humongous amounts of data are communicated over a public infrastructure.

Citizen Journalism: We are all witnessing the rise in a new form of journalism, where non-journalist have an active role in collecting, analyzing and generating newswire based on their personal rules of fairness and objectivity, often also referred to as *Citizen Journalism* [27]. The website *voiceofsandiego.com*, establishes half of its content from its contributing authors [27] and this new way of performing journalism has profound implications on the future of news media. Content in such applications is currently communicated to users through centralized Web sites, which suffer from the same disadvantages as the P2P video sharing application: they require expensive infrastructures and administration, can easily become a bottleneck during periods of high demand and the postings can be censored by the site owners. Additionally, in these applications the update frequency is significantly high, as authors are continuously adding new postings. Therefore crawler-based meta-services, such as Google News, have to continuously crawl the given resources. Finally, it also difficult to organize the information into regional or local news, if the underlying data does not contain this information.

Using a topologically-aware P2P system, besides of overcoming the problems of centralization, would also enable users to more easily retrieve local or regional content. For instance users in Italy might often be interested in local or regional news. In these cases focalizing on content in the ".it" domain might unveil more relevant and interesting content.

In this paper we present the architecture of an Internet-scale middleware that can be used for efficient content-based search and retrieval in a variety of contexts. Our architecture, *pFusion*, is designed to make keyword search efficient in unstructured P2P networks that are geographically diverse. Although the necessity of topologically-aware overlays has been addressed in the context of structured overlays [8], [33], [48], [56], content-based retrieval in such systems is a more challenging task [9], [14].

We consider unstructured P2P networks because they offer a number of important advantages: (i) Unstructured networks are appropriate for content-based retrieval (e.g. keyword searches) as opposed to object identifiers utilized in structured overlays [3], [10], [11], [43], [54]. (ii) An unstructured network imposes very small demands on individual nodes, as it allows nodes to join or leave the network without significantly affecting the system performance. (iii) Finally unstructured networks can easily accommodate nodes of varying power. Consequently they scale into very large sizes and they offer more robust performance in the presence of node failures and connection unreliability.

Unstructured P2P systems have been utilized in a number of file-sharing systems, such as Gnutella [15], Napster [26] and Morpheus [30]. Although the US Supreme Court had ruled on June 26, 2005, that companies enabling such file-sharing systems can be held liable for the widespread copyright infringement of their users, it is important to point out that the P2P technology is not illegal in its own respect. On the contrary, P2P architectures have been the enabling technology behind several legitimate Internet-Scale services, including the popular Internet telephony service Skype [39], the film content distribution network by the entertainment giant Warner bros. [47], and several other services. Thus, P2P services have a viable prospect, given that these systems are utilized in a legal manner.

This paper builds on our previous work in [55], in which we evaluate various content-based search and retrieval algorithms over popular types of overlay networks. In this paper we propose an integrated architecture that combines two major components to efficiently construct and search an overlay network: The *Distributed Domain Name* order protocol, which is an efficient distributed technique for constructing topologically-aware overlay networks, and the *Intelligent Search Mechanism*, which is an efficient distributed technique for keyword query routing. We perform an extensive experimental evaluation using our *pFusion* architecture. Our objective is to *improve the latency, while maintaining the accuracy of the results*. Our results show that the use of topologically-aware overlays minimizes network delays while maintaining high recall rates and low numbers of messages. To evaluate the impact of the overlay we construct realistic P2P networks based on node distributions from the Gnutella network [15] and end-to-end latency information from the Akamai CDN (Content Distribution Network) [1] and the AMP project at NLANR [18].

II. Related Work

In this section we describe systems that have similarities with *pFusion* in their scope and objectives.

In *PlanetP* [11] participating nodes build a global inverted index over the keyword space which is partially constructed by each node. The framework is based on bloom filters [5], which capture the index of some node P_i. These filters are then randomly gossiped across the rest community so that each peer P_j ($P_j \neq P_i$) can perform a membership query on the contents of P_i. Although bloom filters can efficiently be disseminated in a distributed environment, due to the small size of the bit-vector which maintains the filter, the high *churn rate* [9] in P2P systems makes the maintenance of such structures an endeavor task. Note that the churn rate defines the number of individual peers that move into or out of a network over a specific period, thus a high rate might translate into a non-converging maintenance process of the bloom filters. Compared to our framework, PlanetP can lead user queries to the correct answers in less time, given that the filters are in synchrony with the contents of the peers. However, in an Internet-Scale context, this presumption is not easily satisfiable, thus we focus on only local knowledge at the price of an increased cost in getting to the correct answers. We consider the utilization of bloom filters supplementary to our approach and appropriate when the churn rate or the size of the network is limited.

In a different approach, the pSearch [43] system explores semantic spaces by using advanced techniques from the Information Retrieval field. It uses the *Vector Space Model (VSM)* [17] and *Latent Semantic Indexing (LSI)* [17] to generate a semantic space which is then distributed on top of a CAN [34] structured P2P overlay. Since pSearch exploits a distributed form of LSI and VSM, it can support semantic searches, handling cases of *synonyms* and *homonyms*. For instance a search on "sick" might uncover documents which never mention such a term, but which contain terms such as "ill'. The execution of the core ideas in pSearch require, as with PlanetP, some form of global knowledge. In particular, in order to compute the *inverse document frequency (IDF)* utilized by these methods, somebody has to either have access to the complete document collection, or has to have means to sample this collection. In both cases, this is a non-intuitive task given the unprecedented scale of the environments our work considers. Therefore we consider this approach supplementary to our ideas in the case where the P2P environment is small in size or when the provisioning of semantic queries, rather than keyword queries, is at premium.

The YouSearch project [3] at IBM Almaden proposes the deployment of transient peer crawlers to build a search engine that provides "always-fresh" content. The main idea in YouSearch is that each user using the service contributes its host to become a transient crawler. In effect, this results in a network of transient crawlers in which each crawler maintains an "always-fresh" snapshot of a pre-specified list of Web resources. Each crawler also sends a compact index of its crawl (i.e. a bloom filter), to a centralized component at regular intervals. This helps the system to redirect user queries to the crawler that has content relevant to the query, rather than flooding the network with queries. As with PlanetP, this approach might direct the users to the correct resources in less time, as the bloom filters allow efficient membership queries. On the other hand, the deployment of transient web crawlers, is supplementary to our approach. For instance, in a P2P network of personal video sharers, each peer might in the background also crawl data from the WWW, or other P2P networks, and integrate these new resources in the video sharing network. In summary, the main drawback of the YouSearch approach is the central query resolution engine and the construction and maintenance of the bloom filters.

It is important to highlight that all the aforementioned systems do not take into account the underlying network characteristics making them inappropriate for systems that rely on wide-area packet routing. *pFusion* on the other hand alleviates the burden imposed on the underlying physical network by directing the bulk of the traffic to topologically close-by nodes.

In Foreseer [6], the authors propose the deployment of distributed bloom filters which are explicitly updated on changes or additions in the network. Clearly such an approach has advantages and disadvantages compared to ours: more data is transmitted in the network which results in more resource use but potentially better performance. We take the view that only query or query-reply messages should be transmitted, and we attempt to maximize the use of the information thus dissipated.

In Remindin' [44] a query routing technique is proposed to find peers based on social metaphors. In [21], a peer sampling service is proposed to be employed by gossip-based protocols. Additionally, [32] uses small world peer networks for distributed web search. However both approaches establish connections to remote peers based on their query/queryhit patterns while we additionally concentrate on selecting peers based on their topological properties. Finally, an alternative approach for publish/subscribe systems based on structured P2P systems appeared in [16].

Other systems with similar objectives to pFusion include the InfoBeacons [10], PIER [19], and Odissea [42] projects. Note that when each distributed peer returns its own locally highest-ranked answers, then we need to deploy a distributed ranking algorithm that combines the results from the different peers [23], [57]. If such an operation is too expensive, then somebody can focus on the k highest ranked answers, for some user-defined parameter k, using well established distributed top-k query processing algorithms [7].

The remainder of the paper is organized as follows: In Section III we provide an overview of the *pFusion* architecture. Section IV presents the overlay construction module of our architecture, which organizes nodes by taking into account the underlying physical network. In Section V, we introduce the query routing algorithms utilized by the pFusion architecture. Section VI describes our experimental methodology and Section VII the results of our evaluation. Finally, Section VIII concludes the paper.

III. The pFusion Architecture

In this section we provide an overview of the *pFusion* Architecture. *pFusion* is completely decentralized as there is no centralized component that assists in the network construction, maintenance or search process. The architecture of each *pFusion* node (see Figure 2) comprises three basic components:

i) The Distributed Domain Name Order (DDNO) Module (described in section IV), which is a distributed overlay construction module utilized to cluster topologically close-by nodes together. This is achieved by having each node connect to $d/2$ random neighbors and $d/2$ other nodes in the same domain (siblings), where d denotes the number of neighbors. Sibling nodes are efficiently discovered by the deployment of distributed lookup messages and local *ZoneCaches*, which contain information on which domains are reachable in an r-hop radius.

ii) The Intelligent Search Mechanism (ISM), which is a keyword search mechanism used by each *pFusion* node (described in section V). ISM consists of the following two sub-components: 1) A *Profile Mechanism*, which a peer uses to build a profile for each of its neighboring peers (i.e. the query/queryhit pairs) and 2) *RelevanceRank*, which is a peer ranking mechanism that uses the local profiles to select the neighbors that will lead a query to the most relevant answers.

iii) The Local Information Retrieval Engine (LIRE), which is a local index utilized by each node in order to efficiently access its local data repository. Specifically, LIRE organizes local information into disk-based indexes which allow the efficient execution of a wide range of queries (such as Boolean, wildcard, fuzzy and range searches). Note that the indexes in our setting are incrementally updated as new information arrives in the local repository. Merging of query results is performed at the querying node which ranks results on their local score returned by the source. LIRE can also use an external data fetcher for retrieving and storing content pertinent to the interests of the node owner. For example a node participating in a *pFusion* newspaper network may decide to act as a WWW-proxy, similarly to [3], [10], by crawling semi-structured newswire located on regional newspaper websites or RSS [36] feeds which are already available by most major news agencies. The LIRE component is out of the scope of this work.

In designing our architecture we have the following desiderata:

- We focus on unstructured P2P networks, which have been shown to work well for content-based retrieval [3], [10], [11], [43], [54]. Although structured P2P networks have their own important advantages, unstructured networks impose very small demands on individual nodes, and can easily accommodate nodes of varying power. Thus they are compatible with our philosophy of minimizing the requirements the technique imposes on individual peers.
- We focus on fully distributed and autonomous operation for the peers. We try to minimize maintaining any global state or structures that require the active cooperation between peers. Consequently we try to use only local knowledge when we have to decide how a query will be forwarded, or where a peer will connect to the network.
- We leverage previous work [22], [50] that shows how past queries in the P2P system can be cached locally, and used to guide future searches and improve performance. Thus we avoid using mechanisms that have peers exchange information describing their contents, again in accordance with our philosophy of minimizing the dependence of the system to the collaboration of the peers.
- Our objective is to build overlays that can minimize the time and resource requirements for answering keyword queries in unstructured peer-to-peer networks. Nevertheless, the overall recall and precision are still important, and should not suffer for the sake of efficiency.

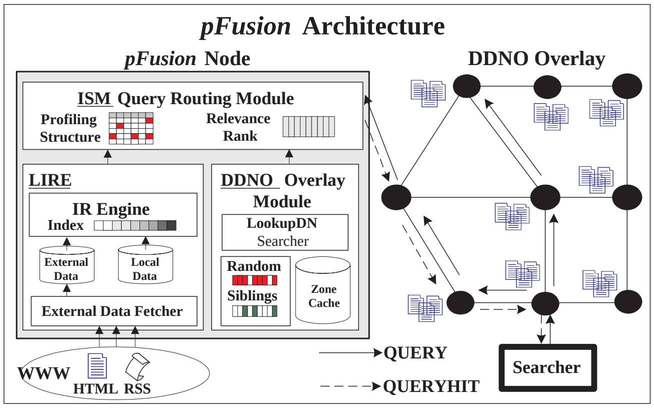

Fig. 2. The *pFusion* Architecture combines two novel components to efficiently search the network: DDNO, which is an efficient distributed technique for creating topologically aware overlay networks, and ISM, which is an efficient and accurate distributed technique for keyword query routing.

IV. The Overlay Module in pFusion

The overlay construction technique that is used by pFusion must be entirely distributed and able to scale well both with the number of nodes and with the rate that nodes join or leave the network.

To achieve our desiderata we create an overlay network where the nodes are well connected to the other nodes in the same domain by making a fixed fraction (one half in the experiments) of the node connections *sibling* connections, while the remaining are connections to random nodes. The two sets of connections serve different purposes: the sibling connections are likely to be low latency connections since they connect nodes in the same domain, so they improve overall efficiency by making local searches very quick. The random connections help to maintain a connected graph. Additionally, these connections enable queries with a small *TTL (Time-To-Live)* to reach a large fraction of the network graph.

A. Alternative techniques for topologically aware overlays

It is important to note that the construction of an optimal overlay is known to be NP-complete [13], [24], [33]. Below we provide a brief description of previously proposed topologically aware overlay construction techniques.

Binning SL Algorithm (BinSL): The Short-Long (SL) topologically aware algorithm, was proposed in [33], and operates in the following way: Each vertex v_i, selects its d neighbors by picking the $d/2$ nodes in the system that have the shortest latency to itself (these connections are called *short links*) and then selects another $d/2$ vertices at random (these connections are called *long links*). Therefore SL is a centralized algorithm as it requires the nxn IP-latency in order to find the latencies between the various node pairs. BinSL is a distributed version of the SL algorithm proposed in [33]. Since the adjacency-matrix of IP latencies is not available in a distributed environment, *BinSL* deploys the notion of *distributed binning* [34] in order to approximate these latencies. More specifically each node uses the round-trip-time (RTT) from itself and k well-known *landmarks* $\{l_1 l_2 .. l_k\}$ on the Internet and each *latency* is classified into $level$ ranges. The numeric ordering of the latencies concatenated by the level values represents the "bin" the node belongs to.

Other Techniques: Recently an approach to create resilient unstructured overlays with small diameters was proposed in [46]. In the proposed algorithm a node selects from a set of k nodes, r nodes at random and then finds from the rest f=k-r nodes the ones that have the largest degree. The algorithm results in networks with power-law distributions of node degrees differentiating it therefore from BinSL and DDNO in which we have a uniform distribution. Topologically-aware overlays have also been addressed in the context of *Structured* P2P overlays in [8], [33], [48], [56]. Systems such as Vivaldi [12] assign synthetic coordinates to hosts so that the Euclidean distance between them estimates the actual network latency. However, the coordinates have to be re-evaluated on an ongoing basis as opposed to DDNO in which sibling nodes are located only during initialization.

Note that the requirement of an efficient overlay is essential in many different types of networks such as Content Distribution Networks (CDNs) [1], Sensor Networks [20], [38] and Mobile Ad-hoc Networks [29], [40]. For instance, the Akamai [1] CDN offers *SureRoute*, which enables users to perform application-layer packet routing, through a virtual overlay network, in order to guarantee the delivery and to improve the performance in wide-area applications. In the context of Sensor Networks, *Data Centric Routing* [20], establishes low-latency paths between the sink and the sensors in order to minimize the consumption of energy. Moreover in *Data Centric Storage* [38] data with the same name (e.g. humidity readings) are stored at the same sensor in the network, offering therefore efficient location and retrieval. Finally in Mobile Ad-hoc Networks, PeopleNet [29] presents a simple, inexpensive and low complexity architecture, for a peer-to-peer wireless virtual social network. PeopleNet exploits the natural mobility of people and their interactions to propagate the queries among neighboring nodes. Additionally the work in [40], studies contact patterns among students in a University campus. The study shows how small inter-contact times between users can be exploited in order to design efficient aggregation algorithms involving only a small number of nodes.

B. Distributed Domain Name Order (DDNO)

In this section we describe the *DDNO* algorithm [53], which clusters nodes belonging to the same domain together without the need of a centralized component. The motivation behind DDNO is to provide a way to find close neighbors without the use of any global infrastructure. We observe that the use of landmarks in the BinSL algorithm, although useful for providing distances between peers, is also restricting since such landmarks have to be identified, and maintained.

Our work aims in extending [33] so that comparable results can be achieved without using any landmarks. Clearly, the key to this is providing a completely distributed way for identifying peers that are likely to be close to a peer that is attempting to join the network. Our approach uses domain names and is motivated by our earlier study on the network traffic of the Gnutella network in [52]. In our study we found that 58% of the nodes in a set of $244,000$ IPs belong to only 20 ISPs. Therefore most nodes have a good probability of finding other *sibling* nodes which makes our scheme beneficial for the largest portion of the network.

Following [33], our technique tries to find, for a new peer, d neighbor peers such that the latency in the resulting overlay is low (assuming shortest path routing). The value of d is an input to the technique. We assume that, although peers can join or leave the network, the number of active peers remains relatively constant, so the required average degree that is required to keep the graph connected can be estimated. Note that our technique is using $d/2$ random connections per peer; this makes the network resilient to peer failures ([46] [4]). Recent work [46] can offer alternative techniques for choosing the value of d but typically this requires additional knowledge on the structure of the network. For example, in [46] nodes find the neighbors of their potential neighbors before choosing their connections.

Peer Domain Names: Each node participating in a DDNO topology has some *Domain Name (dn)*, which is a string that conforms to the syntax rules of RFC 1035 [28]. Such a string, which is case insensitive, can be expressed with the regular expression $dn = label(.subdomain)^+$, where label and subdomain are some strings with certain restrictions, such as length and allowed characters. To determine whether two domain names dn_1 and dn_2 belong to the same domain we introduce two functions: i) *Split-Hash*, which allows us to efficiently encode urls and ii) *dnMatch* which determines whether two domain names dn_1 and dn_2 belong to the same domain or not. The *split-hash* function is a hashing function that splits a domain name dn into k hashes, where k is the number of subdomain strings in dn ($k = |subdomain(dn)|$). A hash function $hash(m, subdomain_{dn}[j])$ is used to hash the $subdomain_{dn}[j]$ using m bits. We chose to use hashcodes instead of raw domain-names because in our technique domain names will be propagated in the network and we want to reduce the size of messages. Two nodes dn_1 and dn_2 have a **dnMatch**(dn_1, dn_2) if the individual hashes match on each subdomain.

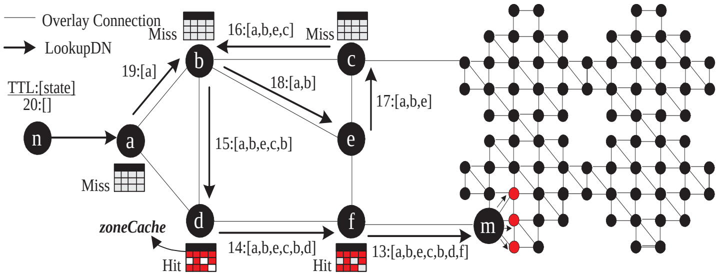

Fig. 3. **Domain-Name Lookup in a DDNO topology.** Each *lookupDN* message retains path information to populate the *ZoneCaches* of other nodes. The list appends shown on the lookupDN message illustrate the accumulated path in ℓ.

Joining a DDNO Topology: Let n denote a node which wants to join an overlay network N. We assume that an out-of-band discovery service (a hostcache or a local cache that stores nodes to which n was connected in some past session) is able to provide n with a random list of active hosts L=$\{n_1, n_2, ..., n_k\}$, for some constant $k \geq \frac{d}{2}$, where d denotes the number of neighbors that n maintains. It is important to notice that the individual hostcaches do not have global knowledge and therefore cannot be used for disseminating some pre-computed overlay structure or the distances between all node pairs to the peers.

After n obtains the list L it first attempts to establish a connection to $d/2$ random nodes, where d is the degree of n. It is quite possible that some or all of the nodes n_i in L are not able to accept any new incoming connections. This might either happen because n_i reached its maximum degree or because n_i went offline. In this case n will need to obtain an additional list L from the hostcache. The next step is to find $d/2$ sibling connections (nodes which have a dnMatch with n). This is achieved by sending a Domain Name `lookupDN` message (described next) to one of the existing (random) neighbors.

Domain-Name Lookup in a DDNO Topology: We now focus our attention on the *lookupDN* procedure which is used by some node n, in order to discover other *sibling* nodes in N. We model the lookupDN message (denoted as ℓ) as a *multicast walker*. The goal of the multicast walker ℓ is to reach some node m that can guide it to the destination (i.e. a sibling of n). Note that before reaching m, ℓ may need to traverse a number of randomly selected neighbors. This can be viewed in Figure 3, in which ℓ takes the random itinerary $[a, b, e, c, b, d]$. At d however, ℓ is allowed to make an informed decision on which neighbor to follow next (in this example node f).

This is achieved by using a special structure coined *ZoneCache* that contains information on which domains are reachable in an r-hop radius (it will be discussed next). At the end of this procedure, ℓ is expected to reach some node m, which is a sibling of n. m then issues a *broadcast* message to all of its own *siblings*. Each of the receiving nodes, including m, will respond with a *LookupOK* message (denoted as ℓ') if they are willing to accept new connections. Therefore node n will end up receiving several answers out of which it will attempt to establish a connection to $d/2$ nodes; these will be n's siblings.

Split-Hash	Neighbor	# Hops	Timestamp
9A78DF	Socket3	3	10000000
421CDE	Socket1	2	10012000
...	...	...	...
2AB356	Socket1	2	10160000

Fig. 4. The **ZoneCache Structure**. It caches domain topological structure information from lookupDN messages that traverse a given node.

One important problem with this approach is that ℓ might get locked in a cycle (e.g. loop $b \to e \to c$ in Figure 3). To avoid this scenario we incorporate state information in ℓ as this also serves as an implicit mechanism to populate the ZoneCaches along ℓ's path. The state information included in ℓ, includes the **split-hash** h on the domain-name of each node that ℓ traversed (i.e. $state_\ell = \{h(v_n), ..., h(v_m)\}$).

ZoneCache: This is a caching structure which is deployed locally at each node and its functionality is to guide ℓ messages to their sibling nodes. In Figure 4 we present a snapshot of the ZoneCache structure. The first column includes the hash of some domain-name and this information is extracted from traversing ℓ messages. The second column indicates the peer connection that will lead a future search (denoted as ℓ_2) to the corresponding destination, and the third column indicates the respective cost in hops. Finally ZoneCache uses a timestamp parameter (fourth column) in order to limit the number of entries in the structure to a total size of C. Once the repository of some node becomes full the node uses the Least Recently Used (LRU) policy to invalidate old entries.

The cache stores only the hashcodes of the nodes that are located within an r-hop radius in order to both limit its size and accuracy. Although neighboring ZoneCaches could actively exchange routing updates at regular intervals, like BGP [31], our passive caching scheme reduces significantly the amount of transmitted message and works well in dynamic environments.

DDNO Topology Maintenance: When a node disconnects from the DDNO topology it does not need to send any a priori notification to the other nodes. However, if some *random* neighbor of n leaves N then n will either attempt to re-establish the dropped connection or find another node from the discovery service outlined before. On the other hand, if some *sibling* of n disconnects then n consults its *ZoneCache* in order to send the new lookupDN message towards a current sibling. It is expected that n will discover another sibling in only 2 hops (as a node already maintains $(\frac{d}{2} - 1)$ siblings).

V. Query Routing in pFusion

In this section we describe the query routing algorithms that can be used to perform content-based searches in *pFusion*. The techniques do not use any global knowledge, thus they are completely decentralized and scale well with the size of the network.

A. Alternative Query Routing Techniques

Below we provide a brief description of alternative query routing techniques evaluated in this paper. BFS is a technique widely used in P2P file sharing applications, such as Gnutella [15]. It works by recursively forwarding the query on each node, to all neighbors (except the sender). In order to avoid flooding the network with queries, as the network might be arbitrary large, each query is associated with a Time-To-Live (TTL) field which determines the maximum number of hops that a given query should be forwarded. In [22] we propose and evaluate the *Random Breadth-First-Search (RBFS)* technique. RBFS improves over the naive BFS approach by allowing each node to forward the search request to only a fraction (0.5 in the experiments) of its peers.

Yang et al. [50] present a technique where each node forwards a query to some of its peers based on aggregated statistics. They compare a number of query routing heuristics and show that the *Most Results in Past (>RES)* heuristic has the best performance. In >RES a peer u forwards a search message to the k peers which returned the most results in the last 10 queries.

B. The Intelligent Search Mechanism (ISM)

Keys to improving the speed and efficiency of the information retrieval mechanism is to minimize the communication costs, that is, the number of messages sent between the peers, and to minimize the number of peers that are queried for each search request. In [22] we propose the Intelligent Search Mechanism (ISM) which is a fast and efficient mechanism for information retrieval in unstructured P2P networks. ISM achieves reduced messaging by having each peer to profile the query/queryhit activity of its neighboring nodes. It then uses this knowledge to forward queries to the neighbors that are most likely going to reply to a given query. ISM consists of two components that run locally in each peer: i) a Profile mechanism and ii) a RelevanceRank function.

i) Profile Mechanism: each node maintains in a repository the T most recent queries and the corresponding queryhits along with the number of results. Once the repository is full, the node uses the Least Recently Used (LRU) replacement policy to keep the most recent queries.

ii) RelevanceRank (RR) is a function used by a node P_l to perform an online ranking of its neighbors in order to determine to which ones to forward a query q. To compute the ranking of each peer P_i, P_l compares q to all queries in the profiling structure, for which there is a queryhit, and calculates $RR_{P_l}(P_i, q)$ as follows:

$$RR_{P_l}(P_i, q) = \sum_{\text{j = "Queryhits by } P_i\text{"}} Qsim(q_j, q)^{\alpha} * S(P_i, q_j)$$

where the deployed distance metric $Qsim$ is the cosine similarity [2] and $S(P_i, q_j)$ is the number of results returned by P_i for query q_j. RR allows us to rank higher the peers that returned more results. α allows us to add more weight to the most similar queries. For example, when α is large then the query with the largest similarity $Qsim(q_j, q)$ dominates the formula. If we set $\alpha = 1$ all queries are equally counted, while setting $\alpha = 0$ allows us to count only the number of results returned by each peer. Note that other numeric similarity metrics, such as Jaccard coefficient, the dice coefficient and the inner product (listed in [37]), are also appropriate in our setting since these can again be computed locally.

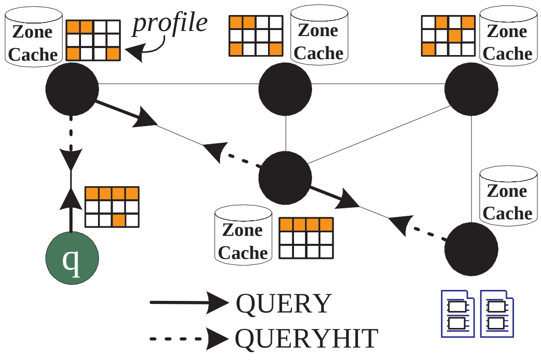

Fig. 5. Searching in a P2P network using ISM. The profiling structure at each node routes queries to nodes with relevant content.

ISM works well in environments which exhibit strong degrees of query locality and where peers hold some specialized knowledge. Our study on the Gnutella network shows that this characteristic is a reasonable assumption [52]. Although we propose the ISM mechanism for keyword-based searches, the basic mechanism can also be used for content-based retrieval of audio, image or video features as long as a similarity function between the queries can be provided. Finally, in [22], [54], we have conducted an analytical study of the RBFS search mechanism.

C. ISM over DDNO

The existence of sibling and random links in the DDNO topology can be further exploited in query routing. If a query is likely to generate query hits in the local domain then the peer can use the ISM mechanism to send the query only to the most likely sibling nodes. We call this mode the *"short"* query mode. In this situation the query is only propagated along sibling connections, and specifically the ones chosen as the most relevant by ISM (see Figure 5),

If the query results are not satisfactory, the query node can then re-issue the query using both sibling and random nodes. However the "short" query mode can be very useful in many situations. Since all messages are sent to sibling nodes (in the same domain), it is very likely to terminate very quickly, especially since it likely needs a small number of hops to explore the domain. In addition, if there is locality of interests, local peers are more likely to have good results. Such a case can easily come up in our distributed newspaper example: most queries are likely to be about local news.

VI. Experimental Evaluation Methodology

To validate that it is possible to obtain the benefits of a topologically aware overlay using only local knowledge, our experimental evaluation focuses on the following three parameters: (i) the *Aggregate Tree Delay* (Δ_T), which is a metric of network efficiency for a given query that spans in

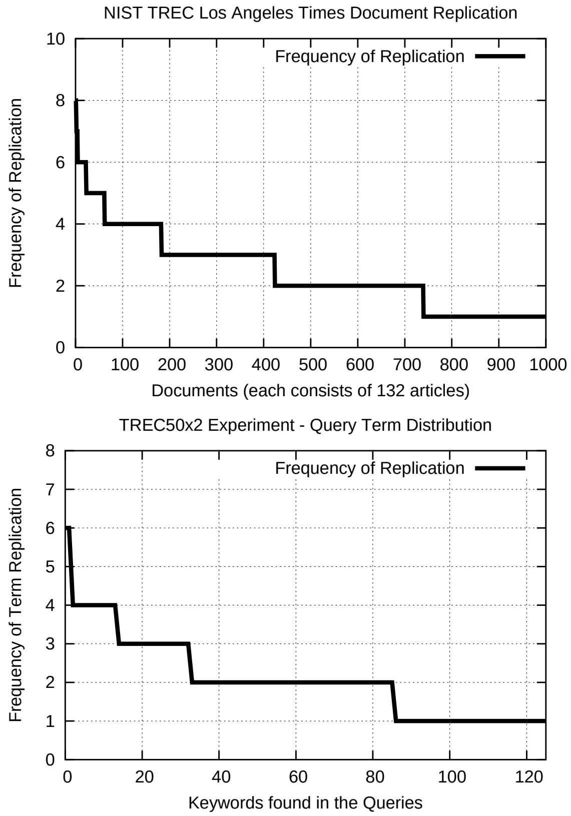

Fig. 6. **a) Document Replication** of the TREC-LATimes dataset, **b) Query Term** distribution for the TREC50x2 queryset

the subgraph G', (ii) the *Recall Rate*, that is, the fraction of documents each of the search mechanisms retrieves, and (iii) the number of *Messages* consumed in order to find the results.

To describe the *Aggregate Tree Delay*, let $G = (V, E)$ denote an overlay graph with a vertex set $V = \{1, 2, ..., n\}$ and an edge set E. Queries posted in G, create a spanning tree T which spans over the sub-graph G' ($G' \subseteq G$). Intuitively, an efficient overlay network can improve the query latency by minimizing the sum of latencies along the explored edges. More formally, the goal of our overlay construction technique is to minimize: $\Delta_T = \sum_{\forall \epsilon \in T} w(\epsilon)$, where w is the latency associated with edge ϵ in the tree T. It is important to notice that the delay cost associated with each edge might be different for each direction between two nodes v_i and v_j (i.e. $delay(v_i, v_j) \neq delay(v_j, v_i)$). This happens because packets on the Internet follow different paths or because the upstream and downstream bandwidth of a node is different (e.g. Cable/ADSL Modem Users).

For the *Recall Rate*, we use as the baseline of comparison the results retrieved by querying the collection in a centralized setting (i.e. as a corpus of documents) which therefore returns all relevant documents. We chose to implement the algorithms that require only local knowledge (i.e. BFS, RBFS, >RES and ISM) over Random, BinSL and DDNO topologies of the same size and degree.

TABLE I
DOMAIN DISTRIBUTION OF THE TOP-10 DOMAINS IN A DATASET OF 244,000 IPS FOUND IN GNUTELLA.

#	Domain	%	#	Domain	%
1	rr.com	10%	**6**	cox.net	3%
2	aol.com	8%	**7**	bellsouth.net	2%
3	t-dialin.net	6%	**8**	shawcable.net	2%
4	attbi.com	6%	**9**	sympatico.ca	2%
5	comcast.net	3%	**10**	optonline.net	1%

A. Dataset Description

We use two series of experiments based on the *TREC-LATimes* [45] dataset, a document collection of randomly selected articles that appeared on the LA Times newswire from 1989 to 1990. The size of this dataset is 470MB and it contains approximately 132,000 articles. These articles were horizontally partitioned into 1000 XML documents, each subsequently indexed using the Lucene [25] IR API. These indexes, which are disk-based, allow the efficient querying of text-based sources using many IR features. We then generate Random, BinSL and DDNO topologies of 1000 peers in which each peer shares one or more of the 1000 documents. We use this scheme in order to provide some degree of article replication (see Figure 6a).

For the evaluation of the TREC-LATimes corpus we will use, as indicated by NIST, the TREC "topics" 300-450. One problem with the provided 150 queries is that the query term frequency is very low and most terms are presented only once. This is not a realistic assumption since studies on real P2P networks (e.g. [52]) indicate that there is a high locality of query terms. Therefore we used the 150 queries to derive the **TREC50x2** dataset, which consists of a set a ="50 randomly sampled queries out of the initial 150 topics". We then generated a list b of another 50 queries which are randomly sampled out of a. *TREC50x2* is then the queries in a and b randomly shuffled and the distribution of query terms can be viewed in Figure 6b.

B. Simulating Network Distances

Evaluating distances in network topologies requires a dataset in which the IP latencies are not synthetic. Therefore we base our experiments on traceroute data from the *Active Measurement Project* (NLANR) [18] and ping data from the Akamai Internet mapping infrastructure [1] (AKAMAI).

i) NLANR: This dataset contains traceroutes between 117 monitors of hosts on the Internet2 backbone. The trace snapshot that we used, was obtained in January 2003 and had a raw size of 1.8 GB. From the initial set of 117 monitors we extracted the 89 monitors which could be reversed DNS (i.e. given their IP we obtained a DNS name). We then construct the nxn IP-latency matrix (for all n=89 physical nodes), that contains the latency among all monitors. Since all 89 hosts are located at different domains, we randomly and uniformly replicate nodes within each domain, providing therefore some degree of host replication per domain.

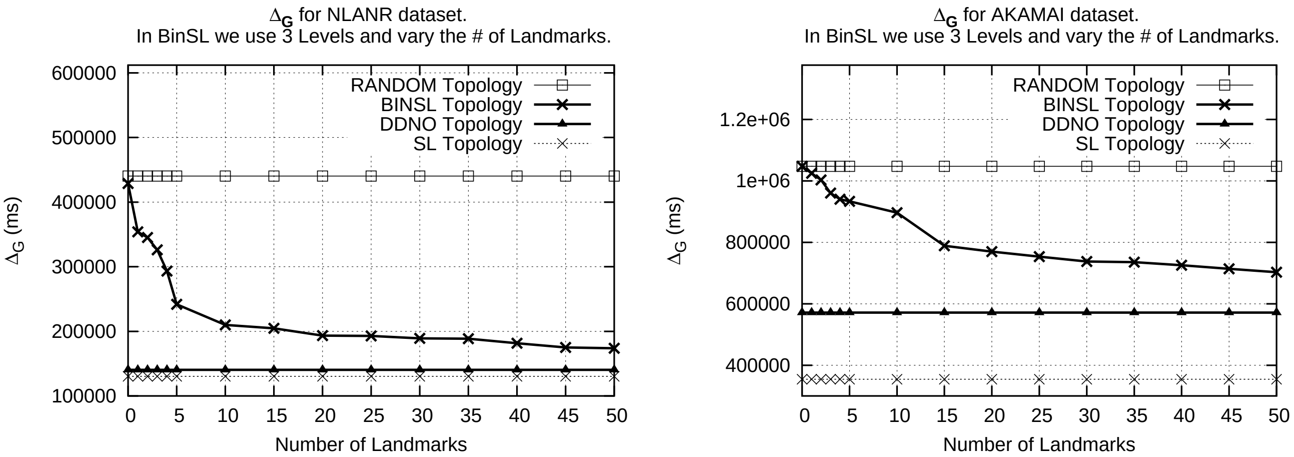

Fig. 7. **Aggregate Graph Delay** (Δ_G) for the NLANR (left) and AKAMAI (right) dataset using four different overlay topologies.

ii) AKAMAI: This dataset contains latency measurements obtained in August 2004, from a very large number of "well-positioned" servers to DNS servers on the Internet. In order to obtain an nxn IP-latency matrix among all DNS servers, we use the triangle inequality to lower bound the desired distances. More formally, let s be a server that pings n DNS servers in the set D. This creates a set of n edges, each with an associated $distance(s, d)$ ($d \in D, \forall d$). The triangle inequality implies that: $distance(d_1, d_2) \leq min(distance(s, d_1) + distance(s, d_2))$

Since the name of each DNS servers was anonymized in our dataset, we utilize a real set of 244,000 domain names that we obtained by crawling the Gnutella Network in [52]. More specifically we uniformly sample out of our initial set of 244,000 IP addresses 1000 unique addresses and then assign these names to the anonymized DNS servers. Note that the distribution of our sample preserves the initial distribution closely (see Table I). Using this setting, nodes in different domains have latencies found in the Internet while nodes in the same domain are randomly assigned latencies in the range [10..50]ms.

C. The pFusion Simulation Infrastructure

In order to benchmark the efficiency of the various information retrieval algorithms over various overlay topologies, we have implemented *pFusion*, using our open-source *Peerware* system [1]. We use *pFusion* to build a decentralized newspaper network which is organized as a network of 1000 nodes. Our experiments are performed on a network of 75 workstations (each hosting a number of nodes), each of which has an AMD Athlon 800MHz-1.4GHz processor with memories varying from 256MB-1GB RAM running Mandrake Linux 8.0 (kernel 2.4.3-20) all interconnected with a 10/100 LAN. *pFusion* is written entirely in Java and comes along with an extensive set of UNIX shell scripts that allow the easy deployment and administration of the system. It consists of 13,500 lines of code with 6500 lines devoted to the core protocol implementation, 5000 lines to the *pFusion* node and 2000 lines to topology generators and other supplementary IO components.

Our experimental testbed consists of three components: (i) *graphGen* which pre-compiles network topologies and configuration files for the various nodes participating in a given experiment, (ii) the *pFusion* client which is able to answer queries from its local XML repository using the Lucene [25] IR Engine, and (iii) *searchPeer* which is a P2P client that performs queries and harvests answers back from a *pFusion* network. Launching a network of 1000 nodes can be done in approximately 10-20 seconds while querying the same network can be performed in around 250ms-1500ms.

VII. Experimental Evaluation

In this section we describe a series of experiments that investigate the effect of the Random, BinSL, and DDNO overlay topology structure on the recall rate and the messaging of the various information retrieval search algorithms discussed in this paper. We focus on investigating if the DDNO topology can significantly minimize the aggregate network delay without sacrificing the recall rate.

Since the possible number of system executions can be very large, due to varying link latencies and the fact that queries might take different paths at the overlay, we present averages over 10 runs. In order to present the statistical significance of our results we additionally present the mean and the 95% confidence intervals[2] for Figure 8 and Figure 9 in Table II and Table III, respectively.

A. Comparing DDNO with other Techniques

In the first experiment, we evaluate the performance of the DDNO topology, and compare it to the BinSL and Random algorithms using the a) NLANR and b) AKAMAI datasets. We evaluate the impact of the number of landmarks on the performance of the BinSL topology. By using more landmarks, the number of *false positives* decreases. This happens because

[1] Available at: http://www.cs.ucr.edu/˜csyiazti/peerware.html

[2] The 95% Confidence Interval indicates that there is a 0.95 probability in any measurement to be within the given interval.

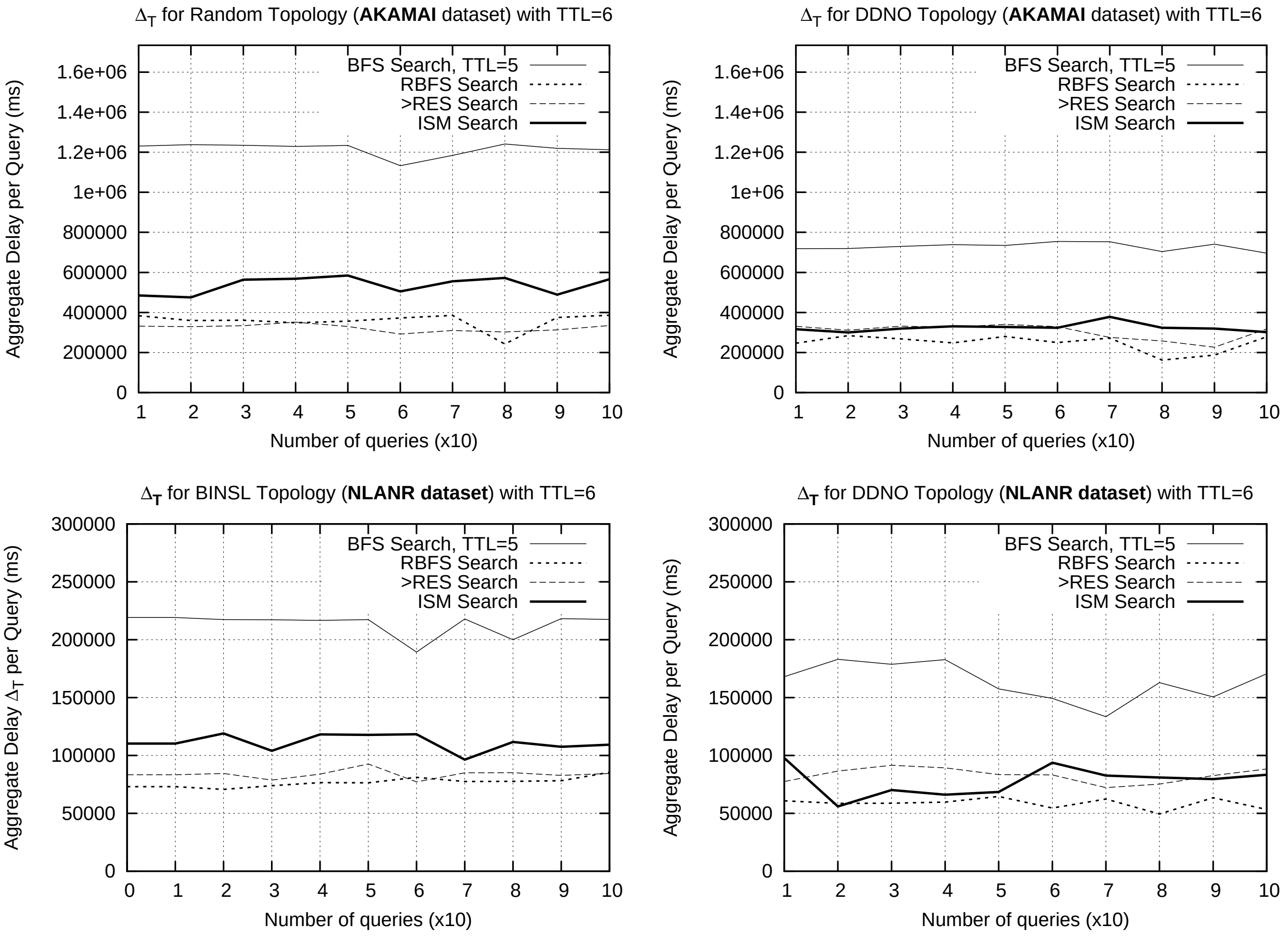

Fig. 8. **Aggregate Tree Delay** for the evaluation of the TREC50x2 queries. In the top row we compare Random (left) and DDNO (right) topologies using the AKAMAI dataset. In the bottom row we compare BinSL (left) and DDNO (right) topologies using the NLANR dataset.

TABLE II
THE MEAN AND THE 95% CONFIDENCE INTERVAL FOR THE PLOTS OF FIGURE 8.

Algorithm	Δ_T **(Random-Akamai)**	Δ_T **(DDNO-Akamai)**	Δ_T **(BinSL-NLANR)**	Δ_T **(DDNO-NLANR)**
BFS	$1,241,618 \pm 51,222$	$698,932 \pm 21,228$	$213,073 \pm 4,603$	$159,699 \pm 9,958$
RBFS	$357,071 \pm 16,900$	$247,667 \pm 14,555$	$77,055 \pm 2,285$	$58,580 \pm 2,305$
>RES	$322,849 \pm 7,871$	$304,757 \pm 11,344$	$83,759 \pm 2,659$	$82,989 \pm 2,837$
ISM	$536,361 \pm 24,822$	$394,160 \pm 24,041$	$111,236 \pm 4,301$	$77,869 \pm 5,787$

we get fewer collisions in the landmark codes of hosts that are not topologically close to each other. We use the centralized SL algorithm as a benchmark to compare against.

In Figure 7, we calculate the sum of the delays w associated with all edges in the respective graphs G (1000 peers each with an average degree of 6). This sum is more formally defined as $\Delta_G = \sum_{\forall \epsilon \in G} w(\epsilon)$, where w is the latency of each edge ϵ in the graph G. We use this metric, instead of the Aggregate Delay Δ_T, as it is independent of the deployed search technique. In BinSL, we first randomly sample out of the original network the set of landmarks. Note that in a real setting, peers would have a predefined list of landmarks (i.e. globally spread HTTP or DNS servers). The figures indicate that by using no landmarks, the BinSL topology is essentially a Random topology. This happens because a node selects all its connections at random which makes Δ_G of the Random and BinSL topologies identical. The figure shows that by adding a few landmarks (i.e. 1-10), Δ_G for the BinSL topology decreases substantially, but after a point Δ_G decreases at a lower rate. Therefore by selecting an arbitrary large number of landmarks may not be very efficient as each landmark probing comes with an additional network cost and because the Δ_G parameter of the graph does not significantly drop. Figure 7 also shows that the lower bound provided by SL is less than what other topologies achieve, but SL is not feasible in practice as it requires global knowledge (i.e. the full

nxn matrix). In the experiments presented in the subsequent subsections, we set the number of landmarks to 20. The reason why the graphs based on the NLANR latencies have a lower Δ_G that those in the AKAMAI latencies, is that NLANR measurements are between Internet2 hosts among which the latency is very low. An extensive experimental comparison of DDNO, BinSL and Random, which includes scalability, failure and bootstrapping experiments, can be found in [53].

B. Minimizing Network Delays

In our second experiment, we investigate if *pFusion* can succeed in minimizing the Aggregate Delay Δ_T of a query, while in the Section VII-C, we will show that this does not affect the recall rate and it also does not increase messaging.

In the top row of Figure 8 we compare the following cases: i) a Random topology with BFS query routing (essentially the Gnutella scenario), ii) a Random topology with an efficient query mechanism (we experiment with ISM, $>$RES, and RBFS), iii) a DDNO topology with BFS query routing, and iv) the *pFusion* architecture that combines a DDNO topology and efficient query routing using the AKAMAI dataset to create the network and answering the TREC50x2 queryset. In the BFS case, we configure each query messages with a TTL parameter of five since this technique is consuming extraordinary amounts of messages. With this setting, query messages are able to reach 859 out of the 1000 nodes.[3] Therefore it was expected that BFS's recall rate would be less than the recall rate obtained by evaluating the whole dataset in a centralized setting. The rest techniques (i.e. RBFS, ISM and $>$RES), use a TTL of 6 as they offer reduced messaging, which allows us to explore the network graph deeper while maintaining low messaging. Finally, the average time to perform a query for the BFS case is in the order of 1.5 seconds but results start streaming back to the query node within the first few milliseconds. By comparing Figure 8 (top-left) with Figure 8 (top-right) can reduce the Δ_T parameter by a factor of three. Figure 8 (bottom-left) with Figure 8 (bottom-right) shows that DDNO also has significant benefits against BinSL. The figures also indicate that the improved search techniques ISM, $>$RES and RBFS have significant savings over the naive BFS approach.

C. Maintaining High Recall Rates and Low Messaging

So far we have seen that by using a DDNO topology we are able to reduce the Δ_T parameter. However this single parameter is not enough in the context of information retrieval applications, as these applications are required to return the most relevant documents. Furthermore, if some search technique always explored the shortest latency neighbors then the Δ_T parameter would be minimal but the query would with very high probability get locked in some region and would not explore the larger part of the network graph. This would consequently reduce the recall rate which is not desirable. In Figure 9, we plot the recall rate required by the different search algorithms. The figures indicate that we can *maintain high levels of recall rate while keeping the Δ_T parameter low* (as shown in the previous subsection).

In the same figures we can also observe the effectiveness of each search technique. More specifically, BFS requires almost 2.5 times more messages than the other techniques. The ISM search technique on the other hand, learns from its profiling structure and guides the queries to the network segments that contain the most relevant documents. On the other hand both RBFS's and $>$RES's recall rates fluctuate, which indicates that $>$RES may behave as bad as RBFS if the queries don't follow some repetitive pattern.

Finally, we note that Random topologies with BFS require slightly more messages ($\approx 10\%$), and consequently are able to score slightly higher recall rates (in our experiments $\approx$0-5$\%$) using the same parameter settings because fewer query paths are *short-circuited*. A query q is short-circuited if its TTL parameter hasn't reached zero, but it is discarded because the same query with a larger TTL already passed from a given node. Note that such a query could explore some additional network segment with its remaining TTL value. However in the *pFusion* approach (DDNO and ISM), the recall rate is not affected because ISM tries to both explore the largest and most relevant segments of the network. This happens because short-circuited areas are penalized by RelevanceRank and explored less frequently.

In addition to the *TREC50x2* dataset, we also experimented with two other querysets: *TREC100*, a set of 100 randomly sampled queries, out of the initial 150 TREC queries, and *TREC10x10*, a set of 10 randomly sampled queries, which are executed 10 times consecutively. In both cases we observe a similar behavior and therefore omit these results for brevity. Note that the TREC100 contains a low locality of reference in its queries, but this does not seem to affect significantly the learning process of the ISM search algorithm.

Finally, we observe that in all curves of Figures 8 and 9, the standard deviation to within 95% confidence, is in most cases 4-7%.

Our experimental results show that random overlay topologies make information retrieval algorithms, proposed in recent literature, significantly more resource demanding and slow. In particular, the Aggregate Tree Delay graphs in Figure 8, shows that random topologies are twice as expensive, in terms of delay along the query path, as the optimized DDNO topology. On the contrary, the minimization of delay that is achieved by DDNO, does not affect the recall rate in such systems, as this is shown by Figure 9.

VIII. Conclusions

We considered and evaluated the impact of the use of topologically aware overlay networks on the performance of fully distributed P2P information retrieval techniques. Specifically, we show that it is possible to efficiently organize the overlay network using only local information, in order to significantly improve the query latency. We also show how to take advantage of this organization when routing the queries in the network. Our experimental results demonstrate that

[3] With a TTL of 6 and 7, we would be able to reach 998 and 1000 nodes at a cost of $8,500$ messages/query and $10,500$ messages/query respectively.

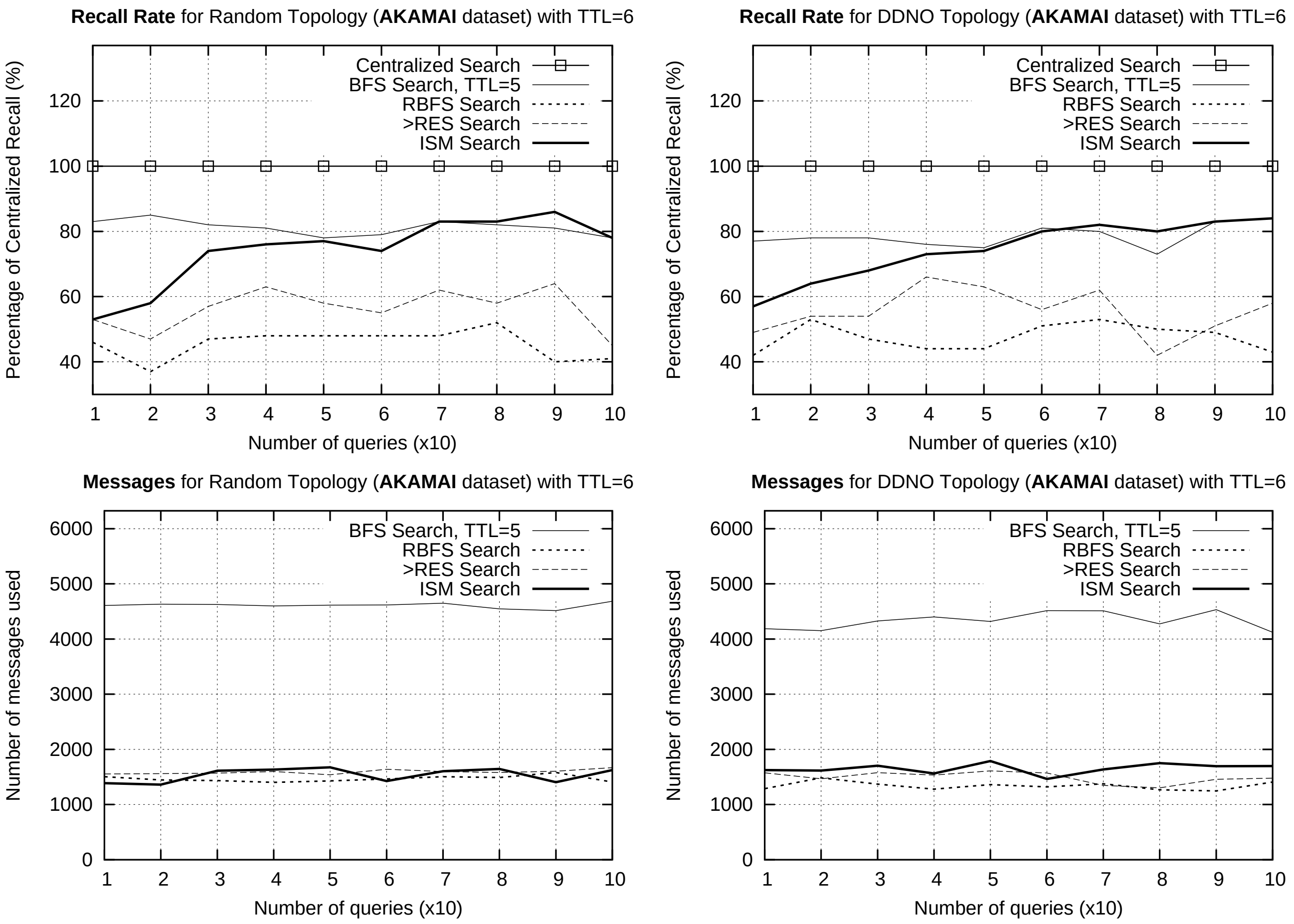

Fig. 9. **Recall Rate** (top row) and **Messages** (bottom row) for the evaluation of the TREC50x2 over Random (left column) and DDNO topologies (right column) using the AKAMAI dataset. For the NLANR dataset we obtained similar results and omitted their presentation due to space limitations.

TABLE III

THE MEAN μ AND THE 95% CONFIDENCE INTERVAL FOR THE PLOTS OF FIGURE 9.

Algorithm	Recall (Rand-Akamai)	Recall (DDNO-Akamai)	Msgs (Rand-Akamai)	Msgs (DDNO-Akamai)
BFS	82.18 ± 5.45	79.11 ± 4.42	$4,603 \pm 217$	$4,133 \pm 137$
RBFS	48.19 ± 5.53	50.29 ± 5.39	$1,409 \pm 49$	$1,450 \pm 82$
>RES	44.66 ± 5.32	59.48 ± 5.54	$1,554 \pm 22$	$1,632 \pm 56$
ISM	75.00 ± 5.96	74.42 ± 5.24	$1,522 \pm 72$	$1,764 \pm 129$

our approach optimizes many desirable properties such as aggregate delays, recall rates and the number of messages. We believe that our techniques are simple to enable seamless integration into existing overlay systems with minimal changes.

ACKNOWLEDGMENTS

We would like to thank Neal Young (UCR & Akamai) and Arthur Berger (Akamai & MIT) for their help in acquiring the AKAMAI dataset, and for fruitful discussions. The data collection was done while the first author was visiting Akamai.

REFERENCES

[1] Akamai Inc., Available at http://www.akamai.com/

[2] Baeza-Yates R.A. and Ribeiro-Neto B.A., "Modern Information Retrieval", ACM Press Series/Addison Wesley, New York, May 1999.

[3] Bawa M., Bayardo R.J., Rajagopalan S., Shekita E., "Make it Fresh, Make it Quick – Searching a Network of Personal Webservers", In *Proceedings of the 12th Int. World Wide Web Conference*, Budapest, Hungary, pp. 577-586, 2003.

[4] Bollobás B., "Modern Graph Theory, Graduate Texts in Mathematics", v. 184, Springer-Verlag, NY, 1998.

[5] Bloom B.H., "Space/Time Trade-offs in Hash Coding with Allowable Errors", Communications of the ACM 13(7), 422-426 (1970).

[6] Cai H., Wang J., "Foreseer: A Novel, Locality-Aware Peer-to-Peer System Architecture for Keyword Searches", In *Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware*, Toronto, Canada, pp. 38-58, 2004.

[7] Cao P. and Wang Z., “Efficient Top-K Query Calculation in Distributed Networks”, In *ACM Symposium on Principles of Distributed Computing*, St. John’s, Newfoundland, Canada, pp. 206 - 215, 2004.

[8] Castro M., Druschel P., Charlie Hu Y., Rowstron A., “Topology-aware routing in structured peer-to-peer overlay networks”, In *IFIP/ACM International Conference on Distributed Systems Platforms*, Heidelberg, Germany, LNCS Vol 2218, 2001.

[9] Chawathe Y., Ratnasamy S., Breslau L., Lanham N., Shenker S., ”Making Gnutella-like P2P Systems Scalable”, In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, Karlsruhe, Germany, pp. 407-418, 2003.

[10] Cooper B.F., “Guiding queries to information sources with InfoBeacons”, In *Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware*, Toronto, Canada, pp. 59-78, 2004.

[11] Cuenca-Acuna F.M. and Nguyen T.D., “Text-Based Content Search and Retrieval in ad hoc P2P Communities”, In *International Workshop on Peer-to-Peer Computing*, Springer-Verlag Vol. 2376, pp. 220-234, May 2002.

[12] Dabek F, Cox R, Kaashoek F, Morris R, “Vivaldi: A Decentralized Network Coordinate System”, In *Proc. of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, Portland, OR, pp. 15-26, 2004.

[13] Garey M.R. and Johnson D.S., “Computers and Intractability: A Guide to the Theory of NP-Completeness”, W.H. Freeman, 1979.

[14] Gao J., and Steenkiste P., “Design and Evaluation of a Distributed Scalable Content Discovery System”, *IEEE Journal on Selected Areas in Communications*, Special Issue on Recent Advances in Service Overlay Networks, 22(1):54-66, January, 2004.

[15] Gnutella.com, Available at: http://www.gnutella.com

[16] Gupta A., Sahin O., Agrawal D., El Abbadi A., “Meghdoot: Content-Based Publish/Subscribe over P2P Networks”, In *Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware*, Toronto, Canada, pp. 254-273, 2004.

[17] Han J. and Kamber M. *Data Mining: Concepts and Techniques, 2nd edition*. Morgan Kaufman, San Francisco, 2006.

[18] Hansen, T., Otero, J., McGregor, A., Braun, H-W., “Active measurement data analysis techniques”, In *Proceedings of the International Conference on Communications in Computing*, Las Vegas, Nevada, Page 105, 2000.

[19] Huebsch R., Hellerstein J.M., Loo B.T., Shenker S. and Stoica I., “Querying the Internet with PIER”, In *Proceedings of 19th International Conference on Very Large Databases*, Berlin, Germany, pp. 321-332, 2003.

[20] Intanagonwiwat C., Govindan R. Estrin D. “Directed diffusion: A scalable and robust communication paradigm for sensor networks”, In Proceedings of the sixth annual international conference on Mobile computing and networking, Boston, MA, pp. 56-67, 2000.

[21] Jelasity M., Guerraoui R., Kermarrec A.-M., van Steen M., “The Peer Sampling Service: Experimental Evaluation of Unstructured Gossip-Based Implementations”, In *ACM/IFIP/USENIX International Middleware Conference*, Toronto, Canada, October 18-20, pp. 79-98, 2004.

[22] Kalogeraki V., Gunopulos D., and Zeinalipour-Yazti D., “A Local Search Mechanism for Peer-to-Peer Networks”, In *11th International Conference on Information and Knowledge Management*, McLean, Virginia, USA, pp. 300-307, 2002.

[23] Karthikeyan Sankaralingam, Simha Sethumadhavan, James C. Browne: “Distributed Pagerank for P2P Systems”, In Proceedings of the *12th IEEE International Symposium on High-Performance Distributed Computing*, Seattle, WA, USA, pp. 58-69, 2003.

[24] Liu Y., Ni L.M, Xiao L., Esfahanian A-H., “Approaching Optimal Peer-to-Peer Overlays”, In *13th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, Atlanta, GA, USA, pp. 407-414, 2005.

[25] Lucene, The Apache Software Foundation, http://lucene.apache.org/.

[26] Napster, Available at: http://www.napster.com/.

[27] Niemann Reports, “Citizen Journalism”, The Niemann Foundation for Journalism at Harvard University, Vol 59. No. 4, pp. 4-33, Winter 2005.

[28] Mockapetris P., “Domain Names - Implementation and Specification”, Request for Comments: 1035, Available at: http://www.ietf.org/rfc/rfc1035.txt

[29] Motani M., Srinivasan V., Nuggehalli P.: “PeopleNet: engineering a wireless virtual social network”, In Proceedings of the 11th Annual International Conference on Mobile Computing and Networking, Cologne, Germany, pp. 243-257, 2005

[30] Morpheus, Available at: http://www.morpheus.com/.

[31] Rekhter Y., Li T., RFC 1771 - A Border Gateway Protocol 4 Request for Comments: 1771, Available at: http://www.ietf.org/rfc/rfc1771.txt

[32] Ruj A. , Wu L.S., Menczer F. “Small World Peer Networks in Distributed Web Search”, In *Proceedings of the 13th international World Wide Web conference*, New York, NY, pp. 396-397, 2004.

[33] Ratnasamy S., Handley M., Karp R., Shenker S., “Topologically-Aware Overlay Construction and Server Selection”, In *The 21st Annual Joint Conference of the IEEE Computer and Communications Societies*, New York, USA, 2002.

[34] Ratnasamy S., Francis P., Handley M., Karp R., and Shenker S., “A Scalable Content-Addressable Network”, In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, San Diego, CA, USA, pp. 161-172, 2001.

[35] Rowstron A. and Druschel P., “Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems”, In *IFIP/ACM International Conference on Distributed Systems Platforms*, Heidelberg, Germany, 2001, LNCS 2218, 2001.

[36] RSS (Really Simple Syndication), Available at: http://purl.org/rss/1.0/spec.

[37] Salton G., “Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer”, Addison-Wesley, Reading, MA. 1989.

[38] Shenker S., Ratnasamy S., Karp B., Govindan R., Estrin D., “Data-centric storage in sensornets”, In *ACM SIGCOMM Computer Communication Review*, Vol. 33 , Iss. 1, pp. 137-142, 2003

[39] Skype, Available at: http://www.skype.com/.

[40] Srinivasan V., Motani M., Ooi W-T. “Analysis and Implications of Student Contact Patterns Derived from Campus Schedules”, In Proceedings of the Twelfth Annual International Conference on Mobile Computing and Networking, Los Angeles, CA, USA, 2006.

[41] Stoica I., Morris R., Karger D., Kaashoek M.F., Balakrishnan H., “Chord: A scalable peer-to-peer lookup service for Internet applications”. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, San Diego, CA, USA, pp. 149-160, 2001.

[42] Suel T., Mathur C., Wu J., Zhang J., Delis A., Kharrazi M., Long X., and Shanmugasundaram K., “ODISSEA: A Peer-to-Peer Architecture for Scalable Web Search and Information Retrieval”, In *6th International Workshop on the Web and Databases*, San Diego, CA, USA, June 2003.

[43] Tang C., Xu Z., and Dwarkadas S., “Peer-to-Peer Information Retrieval Using Self-Organizing Semantic Overlay Networks”, In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, Karlsruhe, Germany, pp. 175-186, 2003.

[44] Tempich C., Staab S., Wranik A., “Remindin’: semantic query routing in peer-to-peer networks based on social metaphors”, In *Proceedings of the 13th international conference on World Wide Web*, New York, NY, pp. 640-649, 2004.

[45] Text REtrieval Conference, National Institute of Standards and Technology http://trec.nist.gov/

[46] Wouhaybi R. and Campbell A. “Phenix: Supporting Resilient Low-Diameter Peer-to-Peer Topologies”, In *The 23rd Annual Joint Conference of the IEEE Computer and Communications Societies*, Hong Kong, China, 2004.

[47] Warner Bros. Entertainment Inc., Available at: http://www.warnerbros.com/.

[48] Xu Z., Tang C., Zhang Z. “Building Topology-Aware Overlays using Global Soft-State”, In *Proceedings of the 23rd International Conference on Distributed Computing Systems*, Providence, RI, USA, Page 500, 2003.

[49] Yahoo Video, Yahoo! Inc., Available at: http://video.search.yahoo.com/

[50] Yang B., and Garcia-Molina H. “Efficient Search in Peer-to-Peer Networks”. In *Proceedings of the 22nd International Conference on Distributed Computing Systems*, Vienna, Austria, pp, 5-14, 2002.

[51] YouTube, Inc. Available at: http://www.youtube.com/.

[52] Zeinalipour-Yazti D. and Folias T., ”Quantitative Analysis of the Gnutella Network Traffic”, Tech. Rep. UCR-CS-2004-04089, Dept. of Computer Science, University of California, Riverside.

[53] Zeinalipour-Yazti D. and Kalogeraki V., “Structuring Topologically-Aware Overlay Networks using Domain Names”, Computer Networks, Elsevier Press, Volume 50, Issue 16 , pp. 3064-3082, 2006.

[54] Zeinalipour-Yazti D., Kalogeraki V., Gunopulos D. “Exploiting Locality for Scalable Information Retrieval in Peer-to-Peer Systems”, Information Systems Journal, Vol. 30, Iss. 4, pp. 277-298, 2005.

[55] Zeinalipour-Yazti D., Kalogeraki V. and Gunopulos D., ”On Constructing Internet-Scale P2P Information Retrieval Systems”, In *Intl. Workshop On Databases, Information Systems and P2P Computing DBISP2P*, VLDB Conference, LNCS 3367, pp. 136-150, Toronto, Canada, 2004.

[56] Zhao B.Y., Duan Y., Huang L., Joseph A.D., Kubiatowicz J.D. “Brocade: landmark routing on overlay networks”, In *First International Workshop on Peer-to-Peer Systems*, Cambridge MA, LNCS Vol. 2429, Pages 34-44, 2002.

[57] Zhu, S. Ye, X. Li. “Distributed PageRank Computation Based on Iterative Aggregation-Disaggregation Methods”, In Proceedings of the 14th ACM International Conference on Information and Knowledge Management Bremen, Germany, pp. 578-585, 2005.

Demetrios Zeinalipour-Yazti is a Visiting Lecturer in the Department of Computer Science at the University of Cyprus. His research interests include Network Data Management, Distributed Query Processing, Storage and Retrieval Methods for Peer-to-Peer and Sensor Networks. He holds a Ph.D. and M.Sc. in Computer Science and Engineering from the University of California - Riverside, and a B.Sc. in Computer Science from the University of Cyprus (2000). He has been a visiting researcher at the network intelligence lab of Akamai Technologies and is currently a reviewer of several scientific journals and conferences in his areas of study.

Vana Kalogeraki is an Assistant Professor at the Department of Computer Science and Engineering at the Univ. of California, Riverside. Her research interests include distributed and real-time systems, peer-to-peer systems and distributed sensor systems. She received her Ph.D. from the Univ. of California, Santa Barbara in 2000. In 2001-2002, she held a Research Scientist position at Hewlett-Packard Labs in Palo Alto, CA. She has published many technical papers, including co-authoring the Object Management Group (OMG) CORBA Dynamic Scheduling Standard and delivered tutorials. She has served as the Program co-Chair of the ”International Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P’03)” at VLDB’2003, the Program co-Chair of the ”13th International Workshop on Parallel and Distributed Real-Time Systems (WPDRTS’05)” and the Program Chair of the ”IEEE Int. Conference on Pervasive Services (ICPS’05)” and the Program co-Chair of the ”10th IEEE International Synposium on Object/component/service-oriented Real-time distributed Computing (ISORC’07)”. She is currently an Associate Editor for the Ad hoc Networks Journal and the Computer Standards and Interfaces Journal. Her research is supported by NSF.

Dimitrios Gunopulos is a Professor at the Dept. of Computer Science and Engineering, in the University of California, Riverside. His research is in the areas of Data Mining and Knowledge Discovery in Databases, Databases, and Algorithms. Dr. Gunopulos has held positions at the IBM Almaden Research Center and at the Max-Planck-Institut for Informatics. He completed his undergraduate studies at the Univ. of Patras, Greece and graduated with M.A. and Ph.D. degrees from Princeton University. His research has been supported by NSF (including an NSF CAREER award), the DoD, the Institute of Museum and Library Services, the Tobacco Related Disease Research Program, and a gift from AT&T. Dr. Gunopulos has served as a PC co-Chair in ACM SIGKDD 2006 and in SSDBM 2003, as a Vice PC-Chair in IEEE ICDE 2004 and in IEEE ICDM 2005, and he is currently an associate Editor at IEEE TKDE and at ACM TKDD.