A Survey of Distributed Search Techniques in Large Scale Distributed Systems

Reaz Ahmed and Raouf Boutaba, *Senior Member, IEEE*

***Abstract*—Peer-to-peer (P2P) technology has triggered a wide range of distributed applications beyond simple file-sharing. Distributed XML databases, distributed computing, server-less web publishing and networked resource/service sharing are only a few to name. Despite of the diversity in applications, these systems share a common problem regarding searching and discovery of information. This commonality stems from the transitory nodes population and volatile information content in the participating nodes. In such dynamic environment, users are not expected to have the exact information about the available objects in the system. Rather queries are based on partial information, which requires the search mechanism to be flexible. On the other hand, to scale with network size the search mechanism is required to be bandwidth efficient.**

In this survey, we identify the search requirements in large scale distributed systems and investigate the ability of the existing search techniques in satisfying these requirements. Representative search techniques from P2P content sharing, service discovery and P2P databases are considered in this work.

***Index Terms*—P2P content sharing, service discovery, P2P databases.**

I. Introduction

NETWORKS of tens or hundreds of thousands of loosely coupled devices have become common in today's world. The interconnection networks can exist in physical or logical dimensions as well as wired and wireless domains. The Internet is the largest distributed system that connects devices through TCP/IP protocol stack. On top of this network there exists many logical overlay topologies, where networked nodes federate to achieve a common goal. Examples of such federations include the Domain Name resolution System (DNS), the World Wide Web (WWW), content sharing P2P systems, world wide service discovery systems and emerging P2P database systems (PDBS). Among these systems, the WWW and the DNS are mature enough and are comprised of relatively static population of Internet hosts (*i.e.*, servers). Content dynamism is also much lower in these two systems, compared to P2P and service discovery systems. Centralized and clustered search techniques (*e.g.*, web crawlers and proxy caches) work well for a network of relatively stable hosts (or web sites) or domain name resolvers. Decentralized (control) and distributed (workload) search techniques are required for a network composed of transient populations of nodes having intermittent connectivity and dynamically assigned IP addresses.

High levels of content and node dynamism in modern large scale distributed systems, including P2P content sharing, service discovery and P2P databases, impose additional requirements on the search mechanism. Flexibility in query expressiveness and fault-resilience of the search mechanism become more important in such environments. The objective of this survey is to investigate the search requirements in large scale distributed systems with a particular focus on three application domains, namely P2P content sharing, service discovery and P2P databases. These three application domains have been exclusively investigated by the research community for the last few years. Hence, we have focused on these three application domains in this survey.

There are surveys on the search mechanisms developed for each of the application domains. For example, surveys on search in P2P content sharing systems can be found in [10], [65], [70], [83], [103]. Comparative studies of different service discovery approaches are presented in [9], [39], [61], [109], while P2P database systems are studied in [50], [57], [78], [95], [106]. We found that the search techniques adopted in these three application domains have similar requirements and functionalities. We therefore believe that a comparative study of their underlying mechanisms can be insightful for future research on distributed search in large scale environments.

The contributions of this paper are as follows: a) We present a survey of existing search techniques in three important application domains, namely P2P content sharing, service discovery and P2P databases; b) We identify the search requirements that are common to these three domains; c) We present a new taxonomy of the search mechanisms and correlate it with existing taxonomies; d) Finally, we present a comparative study of different categories of search techniques in satisfying the search requirements in large scale distributed environments.

The rest of this paper is organized as follows. In Section II we present architectural overview of the three application domains along with the generic advertisement and query model in each domain. Section III illustrates the requirements of a distributed search mechanism. Essential components of a distributed search mechanism and a search taxonomy are presented in Section IV. Representative search techniques from the three application domains are discussed in Section V, Section VI and Section VII. In Section VIII we present a subjective comparison of different categories of search techniques and finally we conclude in Section IX.

Manuscript received 21 October 2009; revised 4 January 2010.

R. Ahmed is with the Department of Computer Science, Bangladesh University of Engineering and Technology, Dhaka-1000, Bangladesh, (e-mail: reaz@cse.buet.ac.bd).

R. Boutaba is with the David R. Cheriton School of Computer Science, University of Waterloo, Ontario, Canada, (e-mail: rboutaba@uwaterloo.ca).

Digital Object Identifier 10.1109/SURV.2011.040410.00097

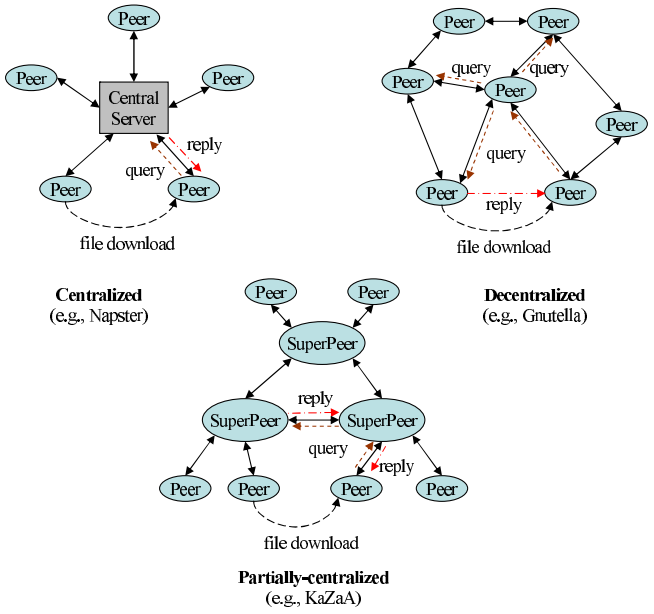


Fig. 1. Content sharing P2P architectures

II. Large-scale Distributed Systems

The goal of this survey is to investigate the decentralized and distributed search techniques for large-scale distributed systems with transient population of nodes. In the following, we will highlight the characteristics of large scale distributed systems in three representative domains: P2P content sharing, service discovery and P2P databases. The identifying properties of these application domains include:

- **Population dynamism**: Transient population of nodes mandates the routing mechanism to be adaptive to failures. Redundant routing paths and replication can improve availability and resilience in such environments.
- **Content dynamism**: Frequent arrival of new contents, relocation (*e.g.*, transfer) of the existing contents and shorter uptime of peers (compared to internet hosts) are the main causes of content dynamism in these systems. Users in these systems often do not have the exact information (*e.g.*, exact filename, or Service Description) about the content they are willing to discover. Rather most of the queries are partial or inexact, which requires the search mechanism to be flexible.
- **Heterogeneity**: In these systems participating population of nodes display wide variation in capacity, *e.g.*, computing power, network bandwidth and storage. This mandates the index information and routing traffic to be distributed based on nodes' capacities.

The rest of this section presents the characteristics of P2P content sharing, service discovery and P2P database systems. We also explain the nature of queries and advertisements in these systems.

A. Peer-to-Peer Content Sharing

Content (*e.g.*, file) sharing is the most popular P2P application. Classifications of the topologies adopted in various P2P content sharing systems can be found in [28] and [70]. In [28], a comparative study of unstructured P2P systems has been presented. Another survey and comparative study on selected approaches from structured and unstructured P2P systems can be found in [66]. In [10], a survey and taxonomy of content sharing P2P systems are presented, while a comprehensive tutorial can be found in [86]. All content sharing P2P systems offer mechanisms for content lookup and for content transfer. Although content transfer takes place between two peers, the search mechanism usually involves intermediate peers. To facilitate effective search, a content is associated with an index file that contains the name, location, and sometimes a description (or keywords) of the content. Search for a content typically involves matching a query expression against the index files. P2P systems differ in how this index file is distributed over the peers (architecture) and what indexing scheme is used (*i.e.*, index structure). From an architectural point of view (see Figure 1), content sharing P2P systems can be *centralized*, *decentralized*, or *partially-decentralized* [10]. **Centralized** P2P systems are characterized by the existence of a central index server, whose sole task is to maintain the index files and facilitate content search. Napster [5] belongs to this category. Centralized P2P systems are highly effective for partial keyword search, but the index system itself becomes a bottleneck and a single point of failure. **Decentralized** architectures remedy this problem by having all peers index their own local content, or additionally cache the index of their direct neighbors. Content search in this case consists in flooding the P2P network with query messages (*e.g.*, through TTL-limited broadcast in Gnutella [2]). A decentralized P2P system such as Gnutella is highly robust, but the query routing overhead is overwhelming in large-scale networks. Recognizing the benefit of index servers, many popular P2P systems today use **partially-decentralized** architectures, where a number of peers (called superpeers) assume the role of index servers. In systems such as KaZaA [3] and Morpheus [4], each superpeer has a set of associated peers. Each superpeer is in charge of maintaining the index file for its peers. Content search is then conducted at the superpeer level, where superpeers may forward query messages to each other using flooding. The selection of superpeers is difficult in such a scheme, as it assumes that some peers in the network have high capacity and are relatively static (*i.e.*, available most of the time). Newer versions of Gnutella [98], [107] also uses this approach.

Advertisements in P2P content sharing systems mostly contain the filename and author-name. Consider the example in Figure 2; a movie file can be advertised as *"The Lord of the Rings - The Two Towers - 2002 (Extended Edition) DVDrip.avi*". For a user it is very unlikely to know the exact name of the advertised file. Rather the user specifies some keywords that may be present in the advertised file name. For example a typical query for the above movie would be *"Lord of the Ring Two Tower"*. Note the keywords *"Ring"* and *"Tower"*; they do not contain the *"s"* as contained in the advertised keywords. This mandates the support for partial keyword matching in P2P content sharing systems.

B. Service Discovery

Service discovery is an integral part of any service infrastructure. A large-scale service infrastructure requires a

Advert.:	*The Lord of the Rings - The Two Towers - 2002 (Extended Edition) DVDrip.avi*
Query:	*Lord of the Ring Two Tower*

Fig. 2. Example advertisement and query in P2P content sharing systems

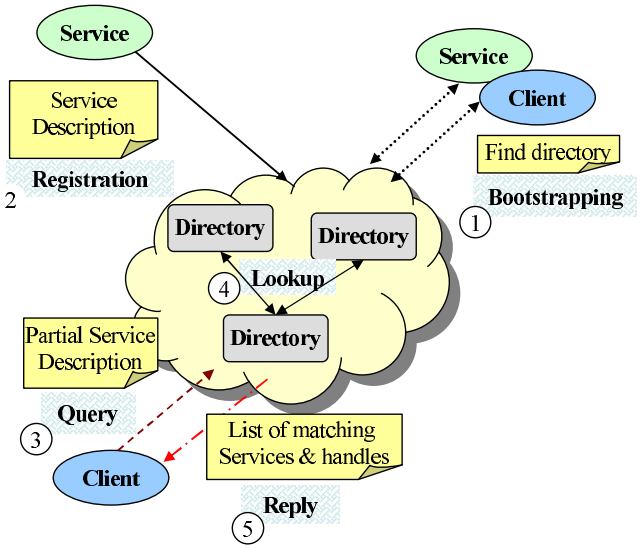


Fig. 3. Service discovery: Generic architecture and steps

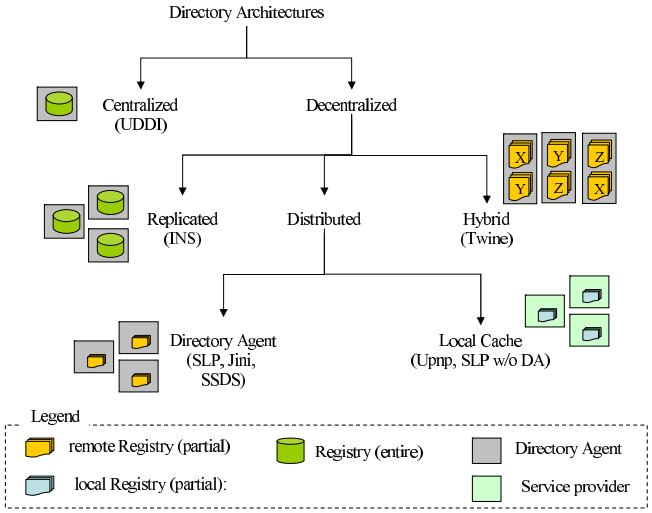


Fig. 4. Taxonomy of the directory architectures

Advertisement	***Query***
Service-type=service:print	Service-type=service:print
Scope-list=staff, grad	*Scope-list=grad*
Location=DC3335	*Paper-size=A4*
color=true	
language=PS	
Paper-size= legal, A4, B5	
URL=diamond.uwatreloo.ca/PCL8	

Fig. 5. Example advertisement and query in Service Discover systems

service discovery mechanism that is open, scalable, robust and efficient. Most of the service discovery systems rely on a three-party architecture, composed of *client*s, *service*s and *directory* entities. Directory entities gather advertisements from service providers and resolve queries from clients. The generic service discovery mechanism can be viewed as a five-step process (see Figure 3) [9]; (1) *bootstrapping*, where clients and service providers attempt to initiate the discovery process via establishing the first point of contact within the system, (2) *service registration*, where a service provider publishes information (a Service Description containing a list of property-value pairs) to a directory entity about the provided service, (3) *querying*, where a client looks for a desired service by submitting a query (usually a partial Service Description) to a directory entity, (4) *lookup*, where the directory entity searches the network of directory entities for all Service Descriptions matching the query and (5) *service handle retrieval*, the final step in the discovery mechanism, where a client receives the means to access the requested service. Some of these steps may be omitted in various discovery approaches. Some of the discovery approaches are based on two-party (client-server) architecture without any directory infrastructure.

Directory architectures adopted by different service discovery approaches can broadly be classified as *centralized* and *decentralized* (see Figure 4) [9]. In a centralized architecture, a dedicated directory entity or registry maintains the whole directory information (as in centralized UDDI [104]), and takes care of registering services and answering to queries. In decentralized architectures, the directory information is stored at different network locations. Decentralized systems can be categorized as *replicated*, *distributed* or *hybrid*. In the replicated case, the entire directory information is stored at different directory entities (as in INS [7]). In the distributed case, the directory information is partitioned, and the partitions are either stored in dedicated directory agents (DA) (as in SLP [41], Jini [99] and SSDS [32]), as per a three-party model or cached locally by the service providers in the system (*e.g.*, UPnP [69] and SLP in DA-less mode), according to a two-party model. Finally, in the hybrid case, the system stores multiple copies of the entire directory information without assigning the entire registry to a single directory entity (as in Twine [14]).

In large-scale networks, a centralized directory becomes a performance bottleneck and a single point of failure. Consistency of the replicas is a major issue in the replicated architecture (like INS), since maintaining consistent replicas is usually bandwidth-consuming. On the other hand, when the directory information is distributed, *e.g.*, partitioned among dedicated directory entities, the failure of one of them leads to the unavailability of part of the directory information. The fully distributed two-party architecture, involving local caches at service and client, attempts to remedy performance bottleneck and single point of failure issues. However, these systems generally do not scale well, since they use multicast-like communication which is expensive in terms of bandwidth. Hybrid architectures seem to offer the best compromise between bandwidth consumption, scalability, and fault-tolerance.

Figure 5 gives an example of a generic advertisement and a query in service discovery systems. In these sys-

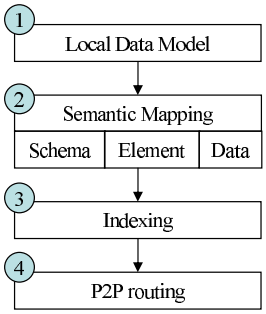
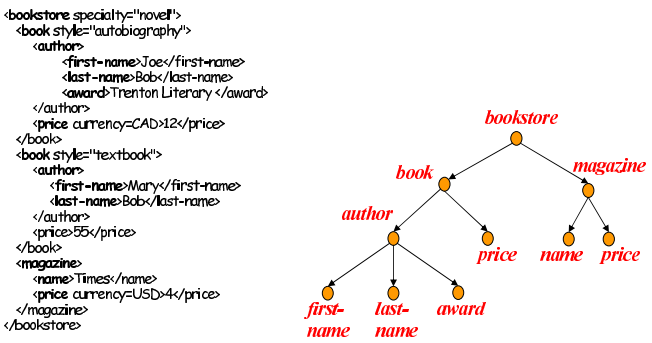


Fig. 6. Functional layers in a PDBS system



(a) An XML advertisement (b) Tree representation

Fig. 7. Advertisement in PDBS

XPath Query

- //author[award]
- /bookstore/book[author/last-name=*Bob*]

Fig. 8. XPath query examples

tems a service is advertised using a list of descriptive property-value pairs, called a *Service Description*. A Service Description typically contains service type (*e.g.*, *Service-type=service:print*), service invocation information (*e.g.*, *URL=diamond.uwatreloo.ca/PCL8*) and service capabilities (*e.g.*, *Paper-size= legal, A4, B5*). In most cases a Service Description is instantiated from a *Service Schema*, which contains meta-information regarding the Service Descriptions for a given class of service (*e.g.*, print service or *service:print*). A Service Schema governs the allowable properties and their types (*e.g.*, string, integer, float, *etc.*) within the Service Descriptions of a given class of services. In most service discovery systems it is assumed that the available Service Schemas are globally known.

Queries in these systems (see Figure 5) usually contain the requested service type and a list of required capabilities of the service (*e.g.*, Paper-size=A4). The list of capabilities provided in a query is a subset of the capabilities list provided in the advertisements it should match against. The result of a query consists of a list of Service Descriptions matching the query.

C. Peer-to-Peer Databases

Peer-to-peer Database Systems (PDBS) have been investigated, more recently, following the success of P2P file-sharing. A P2P database system can be thought of as a data sharing network built on top of a P2P overlay substrate. Search in P2P database systems demands more flexibility than that required by the P2P file-sharing systems. This requirement stems from the existence of semantic (schema) information associated with the shared data. Most of the research works focus on building an additional layer on top of the existing P2P search techniques.

Though PDBSs evolved as a natural extension of Distributed Database Systems (DDBS) [74], they have a number of properties that distinguish them from the DDBS and traditional Database Management Systems (DBMS) [79]. Unlike DDBS, PDBS has no central naming authority, which results into heterogenous schemas in the system. Due to the absence of any central coordination and the large-scale evolving topology, a peer knows about only a portion of the available schemas and data. This mandates a mechanism (*e.g.*, ontology) for unifying semantically close schemas. In DDBS arrival or departure of nodes is performed in a controlled manner, which is not true for PDBSs. Finally, in contrast to DDBS, a peer in a PDBS has full control over its local data.

In PDBS, semantic mapping of schema is a challenging problem. It requires inter-operation between heterogenous data models. XML [96] is used as the de facto standard for this purpose. A survey on the use of XML in PDBS can be found in [58]. In PDBS, XML is used in two ways. Firstly, XML is used for representing data and data models (*i.e.*, schema information). Secondly, XML is used to represent semantic relationships among heterogeneous data models at three different levels: schema level, element level and data level. These levels of granularity also influence the indexing mechanism adopted in these systems.

Figure 6 presents the possible functional layers in a PDBS. Each peer in the system has its own local data model independent of the other peers' data models. The process of translating a local query to other peers' data models is performed by the semantic mapping layer at different granularities, *e.g.*, XML schema mapping, XML element mapping, XML data mapping, *etc*. The third layer is optional, and can maintain indices at different granularities. Finally, the fourth layer is usually one or a combination of the routing mechanisms present in traditional file sharing P2P systems.

Many research work on PDBS assume the existence of an underlying P2P substrate for efficient and flexible query routing, and concentrate on higher level issues including semantic mapping between heterogenous schema, distributed query processing and optimization, *etc*. In this survey, we will consider only those research activities on PDBS that have focused on the issues and challenges related to the query routing mechanism.

Advertisement and query in PDBS are more complicated than that in P2P content sharing and service discovery systems. Figure 7 depicts an example of an XML advertisement which contains information about two books and one magazine. A tree representation of the corresponding *XML Schema* [36] has been presented in Figure 7(b). Analogous to Service Schema, an XML Schema contains meta-information regarding a class

of XML documents. However, the syntax used for describing XML documents and XML Schema are standardized and widely used, compared to the variations in Service Description and Service Schema definition syntaxes used by different service discovery systems.

The most popular query syntax used in PDBS is XPath. Figure 8 presents two examples of XPath queries based on the advertisement presented in Figure 7. The first query finds all *author*s having at least one *award*. The second example finds all books for which *last-name* of the *author* is *Bob*.

III. Distributed Search Requirements

Search is an essential functionality offered by any distributed system. A search mechanism in a distributed system can be either centralized or distributed. For *Centralized Search*, there exists a central core of one or more machines responsible for indexing the contents distributed across the network and for responding to user queries. For networks with lesser degree of dynamism, centralized search mechanisms prove to be adequate. Google [15], Yahoo [6], Alta vista [1] *etc*. are the living examples of centralized search mechanisms, where a set of crawlers running on a cluster of computers index the Webpages around the globe. Compared to the lifetime of the contents shared in P2P networks, Webpages are long lived. Centralized search techniques do not prove to be efficient in large scale distributed systems due to content and node dynamism (as explained in Section II). *Distributed Search* mechanisms assume that both indexing mechanism (analogous to crawlers) and indexed information are distributed across the network. Consequently the design requirements for Distributed Search techniques are different from that for Centralized Search techniques. In the following, we present the most important design requirements for a Distributed Search mechanism.

- **Decentralization**: For a Distributed Search mechanism to be successful, decentralization of control and data are necessary. Decentralization of control refers to the distribution of the index construction process among the participating nodes. There should not be any central entity governing the index construction process in different nodes. Unlike web search engines, the index itself should be distributed across the participating nodes for achieving uniform load distribution and fault-resilience.
- **Efficiency**: The search mechanism should be able to store and retrieve index information without consuming significant resource: mainly storage and bandwidth. In a large scale distributed system advertisements are frequent due to the arrival of new documents and relocation of existing documents. The large user base generates queries at a high rate. This mandates both advertisement and search process to be bandwidth efficient.
- **Scalability**: Efficiency of the search mechanism should not degrade with increase in network size. In addition the number of links per node should not increase a lot with the growth in network size. Join and topology maintenance overhead depends largely on the number of links that a node has to maintain, especially in dynamic environments.
- **Flexibility**: Due to content dynamism, users do not usually have the exact information about the advertised objects. The query semantics offered by the search mechanism should be flexible to support inexact or subset queries. The scalability and efficiency requirements should not be sacrificed for achieving the flexibility requirement.
- **Search completeness**: Search completeness is measured as the percentage of advertised objects (matching the query) that were discovered by the search. Required level of search completeness varies from application to application. A search mechanism should have guarantee on the discovery of rare objects. In the case of popular or highly replicated objects, only a predefined number of matches would suffice for most cases. For specific queries, the number of matching objects would be low and all of them should be discovered by the search. Broad queries, on the other hand, would match a large number of advertised objects. In this case search result may be restricted within a predefined limit to avoid high bandwidth consumption.
- **Fault-resilience**: In large scale distributed systems, participating nodes connect autonomously without administrative intervention. Nodes depart from the network without a priori notification. The search mechanism is expected to advertise and discover objects in a continuously evolving overlay topology, resulting from the frequent arrival and failure of nodes. In many cases index replication and pair-wise, alternate routing paths are used to improve availability.
- **Load distribution**: Heterogeneity in nodes' capabilities, including processing power, storage, bandwidth and uptime, is prominent in large scale distributed systems. To avoid hot spots and to ensure efficiency, the advertisement and search mechanisms should distribute routing, storage and processing loads according to the capabilities of the participating nodes. In other words, uniform distribution of load may result into poor system performance in a large scale distributed system.

In addition to the above mentioned design requirements, a number of other requirements of secondary importance may arise in different scenarios. For example,

- *autonomy* of index placement and routing path selection may be required for security and performance reasons;
- *anonymity* of the advertising, indexing and searching entities may be required in censorship resistance systems;
- *ranking* of search results may be required for full-text search or information retrieval systems; *etc*.

IV. Components of a Distributed Search System

In a large scale distributed system, a distributed search mechanism usually consists of four components as depicted in Figure 9 and presented in the following list.

1) **Query semantics** refer to the expressiveness of a query and the allowed level of semantic difference between queried and advertised information.
2) **Translation** is a function, governing the transformation of semantic information present in a query to a representation that is suitable for query routing.

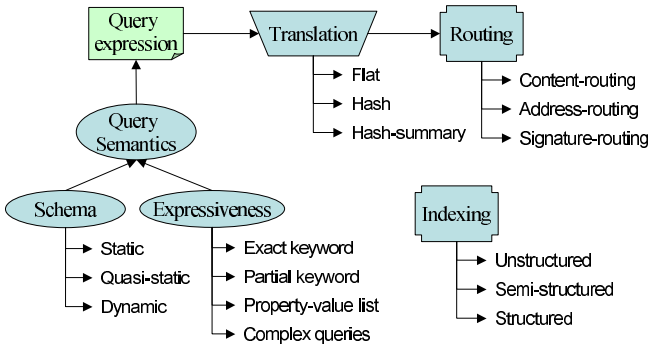


Fig. 9. Components of distributed search mechanism

3) **Routing** refers to the mechanism of forwarding a query to the nodes suitable for answering the query.
4) **Indexing** mechanism determines the distribution and placement of indices (meta information on shared contents) on the overlay network. In many approaches indexing and routing mechanisms are so closely related that it is very hard to separate one from the other.

Each of these components are explained in greater detail in the following subsections.

A. Query Semantics

Any visible (*e.g.*, shared or advertised) object in a distributed system is associated with a set of properties describing the behavioral and functional aspects of that object. Meta information on a set of related properties associated with a class of objects is defined as the *schema* for that class of objects. In a distributed search system, structure and scope (temporal and spatial) of the available schemas influence the query language capability and underlying routing mechanism. The rest of this section highlights two aspects of query semantics: *schema* and *query expressiveness*.

1) Schema: Based on the temporal and spatial scope of the schema, large scale distributed systems can be classified as follows:

- **Static schema**: Most of the file sharing P2P systems have been designed to share one or more specific types of files, *e.g.*, song, movie, software *etc*. For each type of file a specific set of properties is defined that remain unchanged throughout the lifetime of the system. Essentially these systems have one or more static schemas that are globally known.
- **Quasi-static schema**: Most of the service discovery systems fall into this category. Unlike file sharing P2P systems, service discovery systems allow dynamic creation of schema for describing services. Each service instance is advertised as a *Service Description* governed by a predefined *Service Schema* (or template). All schemas in a given service discovery system have to contain a minimal set of predefined *properties* to comply with the specific system under consideration. Though schema can be created dynamically, the rate of such events is very low and the number of available schemas in a given system is much lower than that in PDBS. Furthermore, it is assumed that all the existing schemas in the system are globally known.
- **Dynamic schema**: Most of the PDBSs fall into this category. In these systems heterogeneous schemas exist. Temporal scope of a schema is often bounded by the lifespan of the peer advertising data with that schema. Spatial scope is local to the originating peer and its neighbors; no global knowledge is assumed. Automating the process of semantic mapping between similar schemas is a challenging problem, which may require additional support from the underlying routing mechanism.

2) Expressiveness: Query expressiveness refers to the capability of the query language in expressing information retrieval requirements. Existing research works focus on a wide variety of query expressiveness ranging from simple keyword-based queries to complex queries, such as LDAP filter [45] and XPath [54]. Below is a non-exhaustive list of the different levels of query expressiveness commonly found in distributed search techniques.

- **Exact keyword match** is the minimum level of query expressiveness supported by any search mechanism, and is present in most of the file sharing P2P systems, especially the ones based on DHT[1] techniques. For this level of expressiveness, a globally known fixed schema (with a limited number of properties) is assumed.
- **Partial keyword match** is supported by most of the unstructured techniques as well as some extensions to the DHT techniques. Two major variants in this category can be found. Most extensions to DHT techniques support *partial prefix matching* and unstructured techniques support true *partial matching*.
- **Property-value list** is used by many service discovery techniques. Service Descriptions are specified as a property-value list, and queries are specified as a subset of the advertised property-value list. Most service discovery techniques assume a flat list of property-value pairs and do not support wildcard-based partial matching in property names or values.
- **Complex queries** involve logical and relational operators (*i.e.*, *range queries*), and hierarchical relationships between properties. Complex queries are supported by a few service discovery approaches and most of the distributed XML database systems. For expressing a query, formal query languages, such as LDAP filter [45], XQuery [26], XPath [54], SPARQL [75], RDQL [93] *etc*., are used.

B. Translation

In most distributed systems the query expression specified by a user is not used "*as is*" by the underlying routing mechanism. Instead, the query expression goes through some kind of transformation before it is fed to the routing process. This translation function works as a bridge between user specified queries and the routing mechanism. The *domain* of a translation function is governed by the query semantics as discussed in the previous section. The *range* of a translation

[1]Distributed Hash Tables (DHTs) refer to a class of decentralized search techniques that provide efficient numeric key to node ID lookup service in distributed systems.

function, on the other hand, depends on the routing mechanism used by the underlying overlay. Based on the particular combination of query semantics and routing mechanism, this function can exhibit a wide variation. Translation functions can be broadly classified into the following three categories:

- **Flat**: This type of translation functions do a very little (*e.g.*, filtering) or no change to the query expression and associated semantic information. Such functions are usually used by unstructured and semi-structured indexing mechanisms, and most of the industrial approaches to service discovery.
- **Hash** : Hashing is mostly used by structured and semi-structured search mechanisms. A wide variety of hashing techniques have been proposed for distributed search systems. However, the major problem with this type of translation functions is that they loose semantic information during the hash transformation process. As a result only exact or prefix matching is supported by the search mechanisms that adopt hashing as the translation function.
- **Hash-summary**: This type of translation enables efficient query routing while preserving query semantics. Variants of Bloom filters [18] are the most popular means of representing hash summaries. Hash summaries are mostly used by unstructured and semi-structured search mechanisms.

C. Routing

In overlay networks, routing refers to the process of forwarding a message from a source node to a destination node. The source and the destination nodes are usually at a number of hops away from each other on the overlay. Routing algorithms in overlay networks can be broadly classified into two categories: *uninformed* and *informed*. *Uninformed* routing algorithms do not use the knowledge of query semantics or target node's address in making message forwarding decisions at each hop. *Flooding* [2] [51], *Random walk* [67] and *Iterative deepening* [67] [107] are the representative algorithms in this category. These algorithms are not efficient in terms of generated search traffic, but the robustness is good in highly dynamic environment. Based on the nature of the information used for next hop selection, *Informed* routing algorithms can be classified into the following three categories:

- **Content routing (CR)**: Content routing algorithms utilize the semantic information, embedded in user query for making routing decisions at each hop. Hence, the associated translation function should be from the *flat* category. Content-routing allows partial match and complex queries, but the offered query routing efficiency is low. Moreover, there exists no guarantee on search completeness or the discovery of unpopular objects. Some of the most commonly used content routing techniques are listed below.
  - *Intelligent flooding*: In these techniques a message is selectively forward to some of the neighbors based on some routing knowledge like previous query results, nodes' capacity, type of hosted content, *etc*. Generated message volume is B^{TTL}, where B is the average fan-out and TTL is the time-to-live value.
  - *Hint based routing*: Tentative location of the searched content in the network is used for message forwarding decision at each hop.
  - *Biased walkers*: Fixed number of biased walkers are used in conjunction with routing intelligence like neighbor's capacity, interest, responsiveness, *etc*. Generated message volume is $K \times TTL$, where K is the number of walkers used by the search mechanism.
- **Address routing (AR)**: Address routing is adopted in DHT-based structured P2P overlays, such as Chord [97], CAN [81], Pastry [87] and Kademlia [68]. Different *hash* techniques are used to transform a query into a virtual address on the overlay, and this address is used to route the query to a responsible node. Routing algorithms in this category are efficient in terms of query routing traffic, but they are not appropriate for semantic laden search (*e.g.*, partial matching and complex queries).
- **Signature routing (SR)**: A number of distributed search techniques construct a signature (usually a Bloom filter [18]) [2] of the target object and routes queries based on this signature. These techniques strive to combine the merits of both content-routing and address-routing strategies. Signatures retain (part of or the whole) query semantics and allow information aggregation for efficient indexing. However, search completeness and robustness are not as good as that in address-routing and content routing, respectively.

D. Indexing

Based on *indexing mechanism* and placement of indexed information distributed search techniques can be classified [10] into the following three categories:

- **Unstructured** techniques do not build any index and use uninformed search mechanisms, like Flooding and Random walk.
- **Semi-structured** techniques build index information but do not place any restriction on index placement. Indexed information contains hints on possible location of the content.
- **Structured** techniques rely on some index placement rule that allows one to pinpoint the peer(s) responsible for a given index. Each peer knows the exact location of the contents it has indexed.

V. Search Techniques in Content Sharing P2P Systems

We present various search techniques in this category based on the routing mechanism as follows. Table I summarizes the query semantics, translation functions and routing mechanisms as observed in different search techniques in P2P content

[2]A Bloom filter [18], $B(m, h_1, h_2, \ldots, h_k)$, is an m-bit array that can represent a set, $S = \{a_1, a_2, \ldots, a_n\}$. Here, h_j are hash functions in range $[0, m]$. $B[i] = 0$ for all $1 \leq i \leq m$ if $S = \phi$. $B[h_j(a_i)]$ is set to 1 $(1 \leq j \leq k)$ to insert a_i into B. $c \notin S$ if $\exists j$ $B[h_j(c)]$ equals 0, otherwise $c \in S$ with very high probability.

TABLE I
COMPONENTS OF SELECTED SEARCH TECHNIQUES IN P2P CONTENT SHARING

P2P content sharing					
Ref	**Name**	**Query**	**Translation**	**Type**	**Routing Mechanism**
[64]	Keyword fusion	Multi-keyword	Inverted index	AR	Chord
[49]	Joung et al.	Multi-keyword	Query superset	AR	Chord
[101]	pSearch	Full text, multi-keyword	VSM/LSI	AR	CAN
[16]	Bender et al.	multi-keyword	Hashing+term frequency	AR	Chord
[91]	Squid	Prefix match	Hilbert SFC	AR	Chord
[48]	MKey	Subset match	Query superset	AR	Chord + local flooding
[43]	SkipNet	Prefix match	Flat	AR	Skip List
[30]	Associative Search	Multi-Keyword	Flat	CR	Restricted flooding
[24]	ForeSeer	Multi-Keyword	Flat	CR	Result bias+1-hop flooding
[102]	APS	Partial keyword	Flat	CR	Result bias+random walk
[31]	RI	Document category	Flat	CR	Content bias+random walk
[27]	GIA	Partial keyword	Flat	CR	Capacity bias+random walk
[62]	NSS	Multi-keyword	Bloom filter (BF)	SR	Informed flooding
[82]	PLR	Multi-keyword	Attenuated BF	SR	Hint bias
[59]	EDBF	Partial keyword	Exp. decay BF	SR	Hint bias

sharing domain. In the rest of this section details on these mechanisms are presented.

A. Address Routing Techniques

Majority of the address routing techniques rely on Distributed Hash Tables (DHT). In general DHT-based techniques, like Chord [97], CAN [81], Tapestry [108], Pastry [87], Kademlia [68] are not adequate for supporting flexibility requirement for content sharing P2P systems, which warrant minimum flexibility of partial keyword matching. This inadequacy stems from two reasons. Firstly, DHT-techniques use numeric distance based clustering of hashed keywords which is not suitable for partial keyword matching. Secondly, DHT-techniques cannot handle *common keywords problem* well. Popular keywords can incur heavy load on the peers responsible for these keywords; as a result, the distribution of query load will become unbalanced among the participating peers. Moreover, as studied in [12] and [52], the routing performance degrades significantly in address routing techniques in presence of churn.

Attaining partial matching capability without sacrificing routing efficiency is a challenging problem. In this section we consider the research works that focus on solving some variant of the partial matching problem in distributed environment. We do not consider here DHT techniques, like Chord [97], CAN [81], Pastry [87], Kademlia [68] *etc*., since these techniques focus on achieving efficiency at the cost of flexibility and offer exact matching capability only. Rather we discuss the research works like Twine [14], Squid [91], pSearch [101], *etc*., which strive to extend DHT-functionality for achieving partial matching capability.

Inability to support partial keyword matching is considered a handicap for DHT-techniques. In the last few years a number of research efforts have focused on extending DHT-techniques for supporting partial keyword search. Most of these approaches adopted either of the following two strategies:

- Build an additional layer on top of an existing DHT routing mechanism. The aim is to reduce the number of DHT lookups per search by mapping related keywords to nearby peers on the overlay. This strategy is proposed in a number of research works including [49], [64], [91] and [101] .
- Combine structured and unstructured approaches in some hierarchical manner to gain the benefits of both paradigms. Few research works, including [38], [48] and [100], focus on this strategy.

In [42], a **generic inverted index** for supporting partial keyword matching on top of a DHT overly has been presented. A keyword is translated into an address for routing in two steps. First, the keyword is fragmented into η-grams. Then each η-gram is hashed separately to obtain routing addresses. Each peer stores the keyword(s) it is responsible for, along with the list of document links containing that keyword(s). The hashed η-grams form an inverted index, where an advertised η-gram can be discovered by specifying its hash value. This approach requires $O(\omega \log N)$ hops for advertising a keyword containing ω η-grams, assuming that the underlying DHT network has logarithmic routing efficiency. If the advertised document has many keywords then this approach will incur significant advertisement overhead.

Keyword fusion [64] is an inverted indexing mechanism on top of Chord routing protocol. Supported level of query expressiveness is exact matching on multiple keywords. A document advertised with keywords $\{k_1, k_2, \ldots, k_t\}$ is routed to peers responsible for keys $h(k_1), h(k_2), \ldots, h(k_t)$, where $h(\cdot)$ is the DHT's hash function. To reduce the number of DHT-lookups per advertisement and search, a system-wide dictionary of common keywords is maintained. This dictionary is used to eliminate the common keywords from a query or advertisement, then the query or advertisement is routed using the most specific keyword(s) and filtered at the target peer(s) using the more common keywords specified in the query. In essence the translation function filters out common keywords and then applies hashing. This strategy suffers from two problems. Firstly, the advertisement overhead is significant and proportional to the number of keywords. Secondly, maintaining the global dictionary for common keywords is not suitable for large, dynamic networks.

Joung et al. [49] proposed a distributed indexing scheme, built on a logical, d-dimensional hypercube vector space over Chord routing. In this scheme each advertisement is translated

into a d-bit vector according to its keyword set (similar to Bloom filter construction). They treat d-bit vectors as points in d-dimensional hypercube. No restriction on the mapping of a d-dimensional point to a 1-dimensional key space (required for Chord) has been specified. An advertisement is registered to the peer responsible for the d-bit advertisement vector. A query vector (say Q) is computed in the same manner as the advertisement vector. A query is routed to all the peers in the Chord ring[4] that are responsible for a key (say P_i) that is a superset of the query vector Q. Number of DHT lookups per search and query is significant for this approach.

The work by Joung et al. [49] and the inverted indexing method used in Keyword Fusion [64] represent the two extremes of advertisement and query traffic trade off. In [49], an advertisement is registered at one peer (responsible for the advertised key) and a query is routed to all possible peers that may contain a matching advertisement. On the other hand, in Keyword Fusion [64] an advertisement is registered at all the peers responsible for the advertised keywords and the query is routed to the peer responsible for the most uncommon keyword specified in the query.

pSearch [101] utilizes Information Retrieval (IR) techniques for the query translation process. It is built on top of CAN routing protocol and offers content-based full-text search. Keywords associated with an advertised document (or query) are represented as unit vectors. IR techniques like vector space model (VSM) and latent semantic indexing (LSI) are used to compute a unit vector from the keyword list specified in an advertisement (or a query). Similarity between a query and an advertisement (or between two advertisements) is measured using the dot product of the vector representation of the corresponding advertisement and query. Semantically close advertisements and queries are expected to be translated to geometrically close point vectors in the Cartesian space. Now the semantic point vectors from LSI or VSM are treated as geometric points in the Cartesian space of CAN. CAN partitions a d-dimensional, conceptual, Cartesian space into zones and assigns each zone to a peer. However this mapping technique (from LSI/VSM to d-dimensional CAN space) uses the same dimensionality for LSI space and CAN. Thus it needs to have a priori knowledge of the possible keywords (or terms) in the whole system. In reality there can be thousands of possible keywords, and CAN performance degrades at higher dimensions.

In [16], a query correlation based scheme for Web content search on P2P networks has been proposed by **Bender et al.**. In that scheme an *advertising peer* computes the term frequency for each of the advertised document, and uses Chord to route and store each term in the advertisement, separately. An *indexing peer* on the Chord ring stores peerlist and term frequency per peer for each keyword. To resolve a query, each term in the query is hashed and routed to the responsible indexing peers in the Chord ring. Each of the contacted indexing peers, returns a list of advertising peers with term frequency vector. Finally, the *querying peer* uses these term frequency lists to infer k advertising peers most relevant to the queried terms, and downloads the matching documents from those advertising peers.

Squid [91] has been designed to support partial prefix matching and range queries on top of the Chord routing protocol. It uses Hilbert Space-filling Curve (HSFC) [88] for translating keywords to keys. HSFC is a special type of locality preserving hash function that can map points from a d-dimensional grid (or space) to a 1-dimensional curve in such a way that the nearby points in d-dimensional space are usually mapped to adjacent values on the 1-dimensional curve. Squid converts keywords to base-26 (for alphabetic characters) numbers. A d-dimensional point is constructed from d keywords specified in the query or advertisement. Then a d-dimensional HSFC is used to translate a d-dimensional region (*i.e.*, set of points) specified by the query into a set of curve segments in 1-dimension. Finally, each segment is searched using a Chord-lookup followed by a local flooding. Squid supports partial prefix matching (*e.g.*, queries like compu* or net*) and multi-keyword queries; however, Squid does not have provision for supporting true inexact matching of queries like **net**. Another major problem is that the number of (partial) keywords specified in a query or advertisement is bounded by the dimensionality d of the HSFC in use. Another approach analogous to the Squid mechanism has been presented by **Rosch et al.** in [85]. That approach utilizes Z-Curve (instead of HSFC) on top of CAN network.

MKey [48] is a hybrid approach to keyword search. Architecturally there exists a DHT (here Chord) backbone. A backbone node in the Chord ring works as a head for a cluster of nodes, organized in an unstructured fashion. Search within a cluster is based on flooding. On the other hand, Bloom filter is used as index in the backbone. But DHT techniques do not allow Hamming distance based indexing as required for matching Bloom filters. For allowing pattern matching on Chord, the following strategy is used. Nodes on the Chord ring are allowed to have an ID with at most two 1-bits. An advertisement pattern, say 01010111, is advertised to peers 01010000, 00000110 and 00000001; *i.e.*, DHT-keys are obtained from an advertisement pattern by taking pairs of 1-bits in sequential order from left to right. To construct DHT-keys from a query pattern, say 01010011, only the leftmost three 1-bits are used. In this example the 1-bits at 2^{nd},4^{th} and 7^{th} positions. The DHT-keys are obtained by taking the 1-bit in center position (here 4^{th}) and another bit within the left position (here 2^{nd}) and the right position (here 7^{th}). Hence for the query pattern 01010011, generated DHT-keys are 01010000, 00110000, 00011000, 00010100 and 00010010. Evidently the number of DHT-lookups per search or advertisement depends linearly on the number of keywords and the size of the used Bloom-filter. This can be more inefficient than a generic inverted indexing mechanism for inappropriate parameter settings. Besides, the nodes on Chord ring may become points of performance bottleneck for the system.

There exists only a few non-DHT structured approaches to the search problem in P2P networks. **SkipNet** [43] and **SkipGraph** [13] are prominent among them. Both of these approaches use *Skip List* [76] for routing. A skip List is a probabilistic data structure consisting of a collection of

[4]Chord overlay is often referred to as Chord ring, since the peers in Chord protocol are arranged in a circular linked list like manner.

ordered linked lists arranged into levels. The lowest level (*i.e.*, level 0) is an ordinary, ordered linked list. The linked list in level i skips over some elements from the linked list at level $(i-1)$. An element in level i linked list can appear in level $(i+1)$ linked list with some predefined, fixed probability, say p. Storage overhead can be traded for search efficiency by varying p. Search for an element say Q starts at the topmost level. Level i list is sequentially searched until Q falls within the range specified by current element and next element in the list. Then the search recurs to level $i-1$ list from the current element until level 0 is reached. In both SkipGraph and SkipNet, nodes responsible for the upper level elements of the Skip List become potential hot spots and points of failure. To avoid this phenomena, additional lists are maintained at each level. A multi-level indexing mechanism for keyword search based on SkipNet has been proposed in [94]. However, none of these approaches can efficiently support partial keyword search because the underlying data structure used by these techniques, *i.e.*, Skip List, supports prefix matching only.

B. Content Routing Techniques

In content routing systems objects are identified by keywords. Advertisements and queries are expressed in terms of the keywords associated with the shared objects. Address routing systems, on the other hand, identify objects by keys, generated by applying one-way hash function on keywords associated with an object. Key-based query routing is more efficient than keyword-based query routing. The downside of key-based query routing is the lack of support for partial-matching semantics as discussed in the previous section. Content routing systems, utilizing blind search methods, can support partial-matching queries. But, due to the lack of proper routing information, the generated query routing traffic would be very high. Besides, these techniques do not ensure any guarantee on search completeness.

Majority of the content routing techniques in P2P content sharing networks uses either *Flooding* or *Random-walk*. In the following we present representative solutions from each of these techniques.

1) Flooding-based Techniques: In the original **Gnutella** [2] protocol, time-to-live (TTL)-restricted flooding is used for searching. Since, this type of flooding generates huge query traffic and is not scalable, a number of improvements over the flooding algorithm has been proposed. Representative proposals under this category are presented below.

In *modified-BFS* [51] and *directed-BFS* [107] techniques, the flooding process is restricted with selective fan-out at each node, *i.e.*, at each hop a query message is forwarded to a certain percentage of randomly chosen neighbors. These approaches reduce query traffic volume at the cost reduced query hit rate.

Another variant of the original TTL-restricted flooding, as adopted in [67] and [107], is to gradually increase the TTL value starting from one. This method is analogous to the *iterative depending* or *expanding ring search* algorithms. This routing strategy is suitable for discovering popular and hence densely replicated objects. Moreover, this approach can support user controlled incremental retrieval of search results. This approach will generate higher query traffic volume than flooding if the searched object is not available at a nearby peer.

In **Associative Search** [30], peers are organized based on common interest, and restricted flooding is performed in different interest groups. In **ForeSeer** [24], each peer index information from two sets of peers, based on network proximity and recent query responses. All of these techniques reduce the volume of search traffic to some extent, but none provides guarantee on search completeness.

Routing performance in flooding can be improved by selectively forwarding the query messages to superpeers, *i.e.*, peers that remain online for longer and connects with higher number of regular peers. This concept has been utilized in a number of techniques including **GUESS** [33], **Gnutella2** [98], *etc*. In these systems a regular peer connects to one or more superpeers in the system and the superpeers selectively connect to each other to form a superpeer network. In GUESS a leaf peer submits it query to a superpeer it knows. Then this superpeer gradually forwards the message to its neighboring superpeers until a specified number of matches are discovered. To resolve a query the neighboring superpeers forwards the query to all of its descendent leaf peers. In Gnutella2, the superpeers maintain an index of the contents of its leaf peers. To resolve a query, a superpeer forwards it to the relevant leaf peers based on local index. The superpeer also blindly floods the query within one hop in the superpeer network.

2) Walker-based Techniques: In the overlay network context, the term *walker* refers to a message that actively moves along some *sequence* of nodes within the overly, until a *termination criteria* is reached. There exists a lot of proposals for selecting the *sequence* of nodes followed by a walker. Some of those approaches will be discussed later in this section. Possible *termination criteria* for a walker include:

- successful discovery of the searched content
- failure to discover the searched content within pre-specified number of hops (i.e. TTL or Time-To-Live) and
- explicit termination by the initiating node during an iterative routing process.

Compared to flooding, random walkers have much lower bandwidth requirement and the achieved success rate is also very low. Majority of the walker-based search techniques adopt multiple simultaneous walkers to improve success rate and response time at the expense of network bandwidth.

Random-walkers [67] use the simplest message forwarding mechanism, where an incoming walker message is forwarded to a randomly chosen neighbor. Random-walkers exhibit poor success rate because of their blind forwarding mechanism. A number of research proposals attempt to improve the success rate of the naive random-walk mechanism by introducing intelligent routing mechanisms that utilize the knowledge of network topology, available objects and query keywords.

In Adaptive Probabilistic Search (**APS**) [102], each peer gathers knowledge from query keywords and their results. Each node creates a local index on $< Q_i, nID, \{R_j\} >$, which stores the results $\{R_j\}$ return by the neighbor *nID* against the query keyword Q_i. Whenever a query hit occurs, the walker retraces back to the query initiator updating local indices on

the intermediate nodes. Future query forwarding decisions are made based on this information.

In contrast to the reactive index construction mechanism of APS, the Routing Index (**RI**) [31] mechanism proactively gathers index information from neighboring peers. In the RI protocol, documents are classified into thematic categories and each peer maintains local indices as a list of <*categoryID, linkID, goodness*> triples. Here *categoryID* is the ID of the thematic category. *goodness* is a scalar metric quantifying the quality of results returned by any peer accessible through the link *linkID*. RI has better success rate than APS and random walks, but in RI document creation and update requires local flooding.

In **GIA** [27], network topology and heterogeneity are exploited to bias the walkers. In this approach each node declares its *capacity*, based on its network bandwidth, storage and processing power. This *capacity* value has two-fold impact: firstly, on network topology and secondly, on query forwarding. In GIA, a node's degree is proportional to its capacity, which results into the tendency to achieving a superpeer-based network. During query routing, a walker is forwarded to a higher capacity node with a higher probability. Expected indexing overhead is much lower in GIA compared to APS and RI. GIA will exhibit unbalanced distribution of query load, which is usually good for heterogeneous P2P networks as outlined in Section III.

C. Signature Routing Techniques

A number of P2P search techniques construct signatures or bit-vectors from advertised or queried keywords and use these signatures for the indexing and lookup operations, respectively. Bloom filters are the most commonly used signature construction technique. A comprehensive survey on network applications of Bloom filters can be found in [21]. In general, signature based routing techniques incur lower index overhead due to the compact nature of Bloom filters. These techniques offer flexible query matching capability, which is inherent to the to Bloom filter construction mechanism. In this section we focus on a few representative P2P search techniques in this category.

In Neighbor Signature Search (**NSS**) [62], each peer creates and advertises an index (Bloom filter) representing all of its advertised objects. Each peer indexes Bloom filters from all of the neighboring peers within radius r. To resolve a query, a peer searches the content in peers within radius r based on its local index. If no result is found within radius r, the query is forwarded to a peer $2r+1$ hops away, and the process recurs. Indexing information from peers within r hop neighborhood is expensive. To mitigate storage overhead, two aggregation techniques have been proposed. These methods trade off local index storage and maintenance overhead at a peer with query traffic volume. The first aggregation method performs bit-wise OR of all the indices within radius r. Whenever a query matches a local index it is flooded within the indexing radius r. In the second aggregation method, one index is maintained per link. An index for a link, say L, contains the bit-wise OR of the indices from the peer within radius r that are accessible through link L. If a query matches a local index, then it is forwarded to the associated link. Experimental results presented in this work show that logical OR-based aggregation of Bloom filters is not suitable for indexing information from peers more than one hop away.

In bitwise-OR based aggregation of Bloom-filters, information loss occurs significantly as more Bloom-filters and ORed. To minimize this impact different variants of Bloom-filters have been proposed. We present two such alternates below.

In Probabilistic Location and Routing (**PLR**) [82] each peer stores a list of Bloom filters, named *Attenuated Bloom filter (ABF)*, per link. The i^{th} Bloom filter in the ABF for link L summarizes the resources that are $i-1$ hops away through link L. A query is forwarded through the link with a matching Bloom filter at the smallest hop-distance. This approach aims at finding the closest replica of a document with a high probability. However, index maintenance overhead is high in this approach and convergence is hard to achieve if the peers exhibit high degree of dynamism.

To reduce the impact of peer dynamism, *Exponentially Decaying Bloom Filter* (**EDBF**) has been proposed in [59]. In EDBF, the 1-bits in a Bloom filter decay (*i.e.*, set to zero) with an exponential probability depending on the hop distance from the peer originating the Bloom filter. Each peer gathers advertisement EDBF from its neighbors; constructs its own advertisement EDBF; and advertises it to the neighbors. To construct its own advertisement, a peer resets each of the 1-bits in the received Bloom filter with a constant probability and ORs them with its own Bloom filter. This approach effectively reduces the number of 1-bits (*i.e.*, information content) in an aggregated Bloom filter, but at the cost of an increased probability of false positives.

VI. Search Techniques in Service Discovery

Many service discovery systems rely on a three-party architecture, composed of clients, services and directory entities. Directory entities gather advertisements from service providers and resolve queries from clients. Major protocols for service discovery from industry, like SLP [41], Jini [99], UPnP [69], Salutation [89], etc, assume a few directory agents, and do not provide any efficient mechanism for locating Service Descriptions. Solutions from academia, like Secure Service Discovery Service (SSDS) [32] and Twine [14], target Internet-scale service discovery and face the challenge of achieving efficiency and scalability in locating Service Descriptions based on partial information. A survey on service discovery mechanisms can be found in [9]. A survey on the naming and Service Description schemes used in service discovery techniques and in general in distributed systems can be found in [8]. Another comprehensive survey on the service discovery approaches in global grids can be found in [80].

Table II summarizes the query semantics, translation functions and routing mechanisms for different search techniques in service discovery domain as discussed in the reset of this section.

Secure Service Discovery Service (**SSDS**) [32] arranges directory entities in a tree-like structure and uses hierarchical routing. It uses Bloom filters for translating service descriptions into routing signatures. A bitwise OR-based aggregation

TABLE II
COMPONENTS OF SELECTED SEARCH TECHNIQUES IN SERVICE DISCOVERY

Service discovery					
Ref	**Name**	**Query**	**Translation**	**Type**	**Routing**
[41]	SLP	LDAP filter	Flat	CR	Flooding
[32]	SSDS	Subset/PV-list	Bloom filter	SR	Global hierarchy
[7]	INS	Subtree match	Flat	CR	tree-based flooding
[14]	Twine	Subtree match	Stranding + hash	AR	Chord
[47]	Hu et al.	Service category	Hashing and concatenation	AR	Chord
[90]	Schlosser et al.	Semantic match	Ontology concept $\rightarrow$ d-coord.	CR+AR	2-tier Hypercube
[63]	PWSD	XML path prefix	Stranding + hash	AR	Chord
[92]	Schmidt et al.	Prefix match	Hilbert SFC	AR	Chord

scheme is adopted for reducing the volume of index information at higher level directory entities in the directory tree. In SSDS an advertisement can be discovered by specifying a subset of the advertised property-value list in the query expression. SSDS suffers from load-balancing problem and is vulnerable to the failure of higher level directory entities along the directory tree.

Twine [14] is the scalable version of **INS** [7]. Both INS and Twine use a hierarchical naming scheme. A resource is described using a *name-tree*, composed of the properties and values associated with the resource. Hierarchical relations between properties are reflected in the tree, *e.g.*, while describing the location of a resource, "room no." appears as a child of the "building" in which it resides. INS uses a tree-based flooding protocol while Twine relies on Chord as the underlying routing mechanism. The translation function in Twine generates a set of strands (substrings) from the advertisement or query (which are expressed in XML format), computes keys for each of these strands, and finally uses these keys for the search or advertisement process. The stranding algorithm in Twine is designed to support partial prefix matching within a name-tree. In Twine, the number of DHT-lookups increases with the number of property-value pairs in the advertisement (or query) and consequently the amount of generated traffic becomes high. Load-balancing is another major problem in Twine. Peers responsible for small or popular strands may become overloaded, and the overall system performance may degrade.

Hu et al. have presented another Chord-based service discovery approach in [47]. In that work, the ID space is partitioned into two parts. The higher bits of an ID is generated by hashing the category of the service begin advertised, while the lower bits are obtained by hashing the IP address of the peer itself. For example, if a peer with IP 172.20.23.10 hosts services of category "services.audio.mp3", then its ID will be generated as $ID = hash(services.audio.mp3) \bigodot hash(172.20.23.10)$, where $\bigodot$ is the concatenation operator. This ID construction mechanism essentially clusters the peers, hosting services of same category, along consecutive positions on the Chord ring. The query routing starts with a Chord lookup of the queried service category followed by a local flooding along the Chord ring. Routing efficiency of this approach may degrade if the number of peers hosting same service (i.e., in same cluster) increases.

Web Services (WS) [20] provide a standard way of interoperating between different software applications, running on a variety of platforms and/or frameworks. Universal Description, Discovery and Integration (UDDI) [104] is the de facto standard for WS discovery. Many research activities are devoted to enhancing and overriding the legacy UDDI specification thriving for efficiency, scalability and flexibility in the discovery mechanism. A detailed survey of such activities can be found in [39]. Table III summarizes some of the proposed architectures for WS discovery. Based on the use of WS ontologies, these approaches can be broadly classified as *semantic-laden* and *semantic-free*. Semantic-laden approaches rely on WS ontology mapping techniques like OWL (Web ontology language) [11] or DAML (DARPA Agent Markup Language) [23] for incorporating intelligence to the discovery process, *i.e.*, for intelligently mapping concep-

TABLE III
SUMMARY OF WEB SERVICE DISCOVERY ARCHITECTURES

Centralized	Registry		Authoritative, centrally controlled store of service descriptions, *e.g.*, UDDI registry [104]
	Index		Non-authoritative, centralized repository of references to service providers; see [20] for details. Web crawlers are used for populating an index database.
Decentralized	Federation		Publicly available UDDI nodes collaborate to form a federation and act together as a large scale virtual UDDI registry [84].
	P2P-based	Semantic-laden	In [90] peers are arranged into a hypercube topology [34] and ontology [105] is used to facilitate efficient and semantically-enabled discovery. An agent-based approach is proposed in [71]. It uses DAML [23] representation for ontology and relies on unstructured search techniques.
		Semantic-free	Both [63] and [92] use Chord overlay for indexing and locating service information. [63] extracts property-value pairs from service descriptions and uses MD5 hashing. [92] uses Hilbert Space Filling Curves for mapping similar Service Descriptions to nearby nodes on the Chord ring. These two approaches are similar to Twine [14] and Squid [91], respectively. In [46], another Chord based solution has been proposed. Here, the ID-space is partitioned in numerically ordered subspaces, and each peer in the Chord-ring maintains links to one peer in each subspace in addition to the regular Chord links. In [53], a Gnutella based unstructured approach utilizing DAML-S and standard WS technology has been proposed for Web-service discovery.

tually related terms in queries and advertisements. Semantic-free approaches, on the other hand, do not utilize WS ontology mapping techniques. These approaches are closely related to the traditional service discovery systems. A number of research work in this category rely on locality preserving hash techniques for translating queries to semantically close advertisements.

VII. Search Techniques in P2P Databases Systems

Several research works on distributed XML databases have adopted DHT techniques, such as Chord [97], CAN [81] and Hypercube [90], for routing. A number of these proposals, including [19], [25] and [37], rely on Chord as the underlying P2P substrate, while the hypercube topology has been used in [72].

Table IV summarizes the query semantics, translation functions and routing mechanisms for different search techniques in distributed XML database domain as discussed in the rest of this section.

XP2P [19], uses XML data model for schema representation, and provides support for resolving XPath [54] queries. Any XML document can be represented as a tree, and an XPath query is used to specify a subtree using a prefix-path originating from the root of the document. For supporting partial prefix-path matching, all possible paths, originating from the root, have to be registered with the Chord ring. To reduce the number of paths to be hashed in the Chord ring during the advertisement and query processes, XP2P adopts the fingerprint construction technique presented in [77]. In this technique, the fingerprint of a binary string $A(t) = (a_1, a_2, \ldots, a_m) = a_1 \times t^{m-1} + a_2 \times t^{m-2} + \cdots + a_m$ is computed as $f(A) = A(t)\%P(t)$, where $P(t)$ is an irreducible polynomial. A useful property of the fingerprint function, utilized by XP2P, is that $f(A \odot B) = f(f(A) \odot B)$, where $\odot$ is the concatenation operator.

Galanis et al. [37] presented a framework for supporting XPath queries on top of Chord routing. XPath queries of the form $/a_1[b_1]/a_2[b_2]/\ldots/a_n\, op\, value$ and queries containing relative path operator (*i.e.*, $//$) are supported. Here, a_i is an element in an XML document, b_i is an XPath expression relative to element a_i, op is an XPath operator like $=$ or $<$, and $value$ is an atomic element in the XML document. The core idea is to build a distributed catalog, where a peer in the Chord ring stores all the prefix-paths for a given element in any XML document stored in the network. In other words, if E is an element in some XML files, then the peer responsible for the key $hash(E)$ stores all the absolute paths (*i.e.*, $/a_1/a_2/\ldots/E$) leading to E in any document stored in the network and the contact information of the peers storing those documents. An XPath query of the form $/a_1/a_2/\ldots/a_k//E$ is routed to the peer (say N) responsible for the key $hash(E)$ and the list of all peers containing XML documents matching the query are extracted. Finally the query is forwarded and executed in the corresponding peers.

Bhattacharya et al. [17] have presented a DHT based approach for distributed XML databases. Their approach is similar to the pSearch [101] technique for P2P content search. Similar to pSearch, they have used Vector Space Model (VSM) for constructing the DHT keys from keywords for indexing and searching. Unlike pSearch, their mechanism is independent of the underlying DHT mechanism. In addition, they have proposed a popularity-based adaptive replication mechanism that dynamically maintains the number of replica of an object proportional to its request rate (*i.e.*, popularity). They also proposed a randomized lookup mechanism that routes a given query to a randomly chosen replica of the target object. Adaptive replication together with the randomized lookup mechanism aid in balancing query load.

RDFPeers [25] uses Resource Description Framework (RDF) [60] for document representation and Chord for routing. An RDF document contains many $< Subject, Predicate, Object >$ triples presented in XML format. A triple, say $< S, P, O >$, is stored in three peers (in the Chord ring) responsible for the keys $hash(S)$, $hash(P)$ and $hash(O)$, respectively. For string literals SHA1 hash function is used. For numeric values (in the $value$ component of a RDF-triple), locality preserving hash function is used. A query can be constructed by specifying any of the three components in a triple. In RDFPeers each document has to be indexed at three peers, which results into increased advertisement and update traffic.

Gu et al. [40] have proposed a two tier model for facilitating RDF triple search in structured overlay network. The upper tier of the proposed architecture follows small world network model (SWNM), where each node knows its local neighbors and a small number of randomly chosen distant nodes with a probability inversely proportional to distance. SWNM usually provide small path length between two nodes and large clustering coefficient [56]. In this way, a set of semantic clusters is obtained at the upper layer. Placement of node within a semantic cluster (*i.e.*, in lower tier) is controlled by Chord protocol. To store a RDF triple in the network, the advertisement is first routed to the appropriate semantic cluster in the upper tier, then within the target semantic cluster it is stored in two places as determined by hashing the $< Subject, Predicate >$ and $< Predicate, Object >$ pairs of the advertised RDF triple.

PeerDB [73] uses an agent-based framework on top of unstructured P2P overlay to achieve distributed data sharing. To accommodate heterogeneity in schema definitions from autonomous peers in the system, PeerDB associates keywords as synonyms with each schema and elements under that schema. These keywords are used as a means of semantic mapping and finding semantically similar schemas. PeerDB uses flooding as the underlying search mechanism. Mobile agents are blindly sent to neighboring peers and a query is executed locally at the each peer, which helps in reducing the volume of network traffic.

JXTA [22] routing has been used by **Kim et. al** [55]. In JXTA architecture a loosely-consistent distributed hash table (LHDHT) is maintained by a set of special peers called Rendezvous peers. Each rendezvous peer maintains a list of known Rendezvous peers and the range of keys associated with each of them. Query routing is performed based on the local information at each Rendezvous peer. In [55], a fixed global schema has been used, whereas existence of heterogenous schema is allowed in [35].

TABLE IV
COMPONENTS OF SELECTED SEARCH TECHNIQUES IN PDBS

P2P databases					
Ref	**Name**	**Query**	**Translation**	**Type**	**Routing**
[19]	XP2P	XPath(absolute)	Fingerprint	AR	Chord
[37]	Galanis et al.	XPath(relative)	XML element hash	AR	Chord
[25]	RDFPeers	Partial RDF triple	RDF element hash	AR	Chord
[17]	Bhattacharya et al.	Multiple-keyword	VSM	AR	DHT
[40]	Gu et al.	Partial RDF triple	Hashing and concatenation	AR	SWNM + Chord
[73]	PeerDB	SQL	Synonym/flat	CR	Flooding
[44]	Humboldt Discoverer	SPARQL/RDF	URI-hash+Flat	AR+CR	Chord+ Controlled Flooding

A hybrid technique, named **Humboldt discoverer**, has been presented in [44]. RDF [60] has been used for describing an advertised resource. SPARQL (Simple Protocol and RDF Query language) [75] has been used for constructing query expressions. SPARQL is a query language for RDF documents that allows formation of complex queries involving relational and logical operators. Routing is done using a three tier architecture, where peers are classified as bottom, middle or top tier peers. Bottom tier peers provide information sources. These peers are clustered into many groups based on the similarity of used ontologies. A middle tier peer is responsible for an ontology and manages a single cluster of bottom tier peers. Middle tier peers advertise their existence to top tier peers, which are organized in a Chord ring and are addressed by the hash of the URIs of the ontologies. In effect, middle tier peers covering the same ontology are grouped under the same top level peer. To resolve a query, all the required ontologies are first determined. For a given ontology, the set of responsible middle tier peers can be reached through the top tier Chord network. Finally, the query is forwarded to each of the middle-tier peers that are responsible for the ontologies used in the query.

VIII. COMPARISON

In this section we compare the capabilities of different search techniques, as discussed in Sections V, VI and VII, in satisfying the search requirements presented in Section III.

Indexing and routing mechanisms are the key factors determining the performance and expressiveness of a search mechanism. For this subjective comparison we classify the search techniques into nine categories based on the indexing and routing mechanisms, as outlined in Table V. In the rest of this section we consider the categories in Table V against the search requirements.

TABLE V
ROUTING MECHANISM VS. INDEXING MECHANISM

	Content	*Signature*	*Address*
Unstructured	Gnutella [2], SLP [41], INS [7], PeerDB [73]		
Semi-structured	Associative [30], ForeSeer [24], APS [102], RI [31], GIA [27]	NSS [62], PLR [82], EDBF [59], SSDS [32], GIA [27]	FreeNet [29], JXTA [22]
Structured	SkipNet [43], SkipGraph [13]		Keyword Fusion [64], Joung et al. [49], pSearch [101], Bender et al. [16], Squid [91], MKey [48], Twine [14], Hu et al. [46], PWSD [63], XP2P [19], Galanis et al. [37], RDFPeers [25], Gu et al. [40]

A. Decentralization

Decentralized index construction process and distributed index maintenance are necessary for the success of any distributed search technique. All of the P2P content sharing systems discussed in this survey exploit decentralized search techniques. Among the service discovery techniques SSDS shows lower level of decentralization since the root and higher level nodes in the indexing hierarchy become central components and points of failure. Most of the discussed approaches in PDBS domain have adopted decentralized search techniques.

B. Efficiency

Network bandwidth is considered to be the most critical resource in each of the three application domains. Overall success of a distributed search technique is determined by its bandwidth efficiency during the search and the advertisement processes. In general Address routing is more efficient than signature routing, which is more efficient than content routing. However performance of address routing based approaches degrades while supporting inexact matching queries, which is essential for all of the three application domains. As presented in Section V many search techniques in P2P content sharing networks including keyword fusion [64], pSearch [101], Squid [101] and MKey [48], aim to support inexact matching queries by transforming an inexact query to a set of exact queries. To resolve each of these exact queries a number of DHT-lookups have to be performed. Thus query routing performance in these systems highly depends on the nature of the query under consideration. A similar situation arises for the address routing based techniques in the service discovery domain (e.g., Twine [14], PWSD [63], Schmidt et. al [92] *etc*.) and PDBS domain (XP2P [19], RDFPeers [25], Galanis et al. [37] and Gu et al. [40]).

Compared to content routing techniques, signature routing techniques using different varieties of Bloom filters have lower indexing and search overhead. This is because these

techniques can store and transmit more information due to the compact nature of the Bloom filter.

C. Scalability

A good distributed search technique should autonomically adopt to changes in network size without degrading routing performance and search completeness. Address routing based approaches including, Keyword fusion [64], Bender et al. [16], Squid [91], PWSD [63], Twine [14], RDFPeers [25], XP2P [19], *etc*., are expected to provide complete search results regardless of network size, but their search traffic may increase significantly with the growth in network size, since these techniques require multiple DHT-lookups per query. CAN and Hypercube based approaches, i.e., pSearch [101] and Joung et al. [49], respectively, will exhibit degraded performance in large networks due to their requirement of higher dimensional structures in larger networks. On the other hand Chord-based techniques will suffer from increased churn rates of larger networks. Chord's limitation in presence of churn is due to its one-way routing table and finger table maintenance overhead. In these cases, Kademlia routing protocol can be adapted instead of Chord. Kademlia has a two-way symmetric routing table and its routing table is automatically updated during the query or advertisement routing process without any extra overhead.

Scalability is a major issue in signature and content routing techniques, due to the indeterminacy of their routing mechanisms. In these systems search completeness can not be guaranteed. Success probability, i.e., probability of finding at least one result, is proportional to the percentage of visited peers and replication factor. Thus in large networks these systems will perform poorly and many queries will fail despite the presence of a matching result somewhere in the network.

D. Flexibility

As depicted in Figures 2, 5 and 7, inexact or similarity based matching amongst advertised and queried information is essential in all of the three application domains considered in this survey. However, existing routing techniques do not offer efficient mechanism for inexact matching in distributed systems. DHT-based techniques have sub-linear relationship between network size and routing cost. However, these systems support exact match queries only. Systems using DHT-techniques for supporting inexact query use an additional conversion layer in order to transform each similarity matching query to more than one exact queries. For example, Squid [91] uses HSFC, pSearch [101] uses LSI/VSM, XP2P [19] uses fingerprinting, *etc*. This conversion mechanism trades-off query semantics for routing efficiency.

Signature routing and content routing techniques, on the other hand, retain semantic information within the query string and use this information for routing. Majority of the signature routing and content routing based approaches use either semi-structured or unstructured indexing, resulting into poor routing performance in large networks. SkipNet [43] and SkipGraph [13] are the only approaches in these categories offering structured indexing over content routing. These two content routing approaches are based on SkipList [76] and support prefix matching only. Moreover, peer join and neighbor link maintenance overhead in these two systems is high, making them inappropriate for large overlay networks.

In summary, inexact matching is necessary in large overlay networks due to high population and content dynamism. But none of the existing search techniques provide satisfactory solution for both efficient routing and inexact matching capabilities. A good solution should adopt structured indexing and content or signature routing techniques for supporting efficient routing and inexact matching queries, respectively. In addition, the solution should have low network overhead for join, leave, failure recovery and link maintenance.

E. Search Completeness

Perception of search-completeness varies in structured and unstructured search techniques. Structured search techniques strive to discover all of the advertised objects matching a query. Unstructured and semi-structured techniques, on the other hand, focus on the discovery of at least one matching result, even if the searched object is rare and is not well replicated in the network. Very high levels of search completeness are achievable in DHT-based structured routing approaches. Since the location of an index within the network is well specified, all of the matching indices can be discovered with high routing efficiency.

Content routing and signature routing based search methods use unstructured or semi-structured indexing mechanisms, which cannot deterministically store and locate the indices within the network. In these approaches, the probability of discovering an object depends on the percentage of visited peers and the level of replication. Hence for large networks, these systems cannot provide complete search results. For discovering rare objects these systems generate a huge query traffic, making them inappropriate for large networks.

F. Fault-resilience

A good distributed search mechanism should function uninterruptedly in a continuously changing overlay topology. Replication and redundant routing paths are necessary for ensuring resilience to peer failures. Structured indexing techniques, including Squid [91], pSearch [101], SkipNet [43], etc., exhibit poor fault resilience due to a number of reasons. Firstly, structured indexing techniques impose strict restrictions on index placement within the overlay, which incurs high index maintenance overhead during node join and leave/failure. Secondly, constraints for neighbors selection are strictly defined in structured indexing techniques, which makes neighbor link maintenance overhead significant in presence of frequent arrival and departure of nodes.

Compared to structured indexing techniques, semi-structured indexing techniques have relaxed constraints on neighbor selection and index placement. As a result these techniques can adapt to population dynamism more easily than structured indexing techniques. Unstructured indexing techniques are the most resilient to population and content dynamism. Since there exists no restriction on neighborhood selection and index placement, topology maintenance overhead is minimal in these networks. Peers can join or leave the

overlay without hampering regular operation of the system. However, the resilience in unstructured indexing techniques is achieved at the cost of reduced routing efficiency and search completeness.

G. Load Distribution

To exploit the heterogeneity in large distributed systems, it is required to distribute the load proportional to the participating peers' capabilities. In structured address routing techniques, index placement within the overlay is strictly defined by the routing mechanism, which makes it expensive and infeasible to dynamically adjust load distribution among the available peers. For example, in pSearch [101], Squid [91] and Twine [14], nodes responsible for common keywords or popular attribute-value pairs may become heavily loaded and choke the performance of the system. Since key assignment is not performed according to the capabilities of the nodes, these systems may suffer from performance problems in heterogeneous environments.

Semi-structured and unstructured indexing techniques adopting content or signature routing mechanisms can successfully exploit the heterogeneity in the participating nodes to improve search performance. For example GIA [27], APS [102] and PLR [82], use neighbors' capabilities like connection bandwidth, stored index size, responsiveness etc., to make routing decisions at each hop and improve over blind routing mechanisms, like flooding and random walk.

IX. Conclusion

In this work we have surveyed the prominent search techniques in three application domains, namely, P2P content sharing, Service Discovery and PDBS. These domains exhibit the same characteristics of high content volatility and population dynamism. The majority of the search techniques in these three domains focus on bandwidth efficient routing mechanisms for enabling semantic-aware and flexible search. Proper combination of indexing and routing mechanisms is essential for achieving bandwidth efficiency and expressiveness, within the same system.

DHT-based address routing techniques utilizing structured indexing mechanisms provide highest level of bandwidth efficiency, but these techniques do not support partial matching between advertisements and queries. On the other hand, content routing techniques utilizing unstructured routing mechanisms deliver the highest level of expressiveness, but exhibit very low bandwidth efficiency. Semi-structured indexing techniques do not provide any guarantee on search completeness, yet they offer moderate level of query expressiveness.

The combination of structured indexing and signature routing mechanisms seems to be a promising candidate for answering the efficiency and flexibility requirements. The reason behind this assertion can be explained as follows. Structured indexing techniques provide guaranteed bandwidth efficiency by pinpointing the location of a content in the network, while signature routing uses the semantic information in query expression for making intermediate routing decisions and message forwarding.

Over the last few years a large body of research works has explored the issues related to distributed search and a number of alternate solutions have been proposed. Each of these solutions has its own merits and demerits, but none of them is satisfactory with respect to bandwidth efficiency and query expressiveness requirements, simultaneously. As explained in Table V, all possible combinations of indexing and routing mechanisms have not been explored yet. Hence, future research in distributed search should focus on unveiling the unexplored alternatives in order to realize the promise for an *efficient* and *flexible* distributed search in large scale distributed systems..

ACKNOWLEDGMENT

This work was supported in part by the Natural Science and Engineering Council of Canada (NSERC) and in part by the WCU (World Class University) program through the Korea National Research Foundation funded by the Ministry of Education, Science and Technology (Project No. R31-2008-000-10100-0).

References

[1] Alta Vista website, [Online]. Available: http://www.altavista.digital.com/.
[2] The Gnutella website, [Online]. Available: http://www.gnutella.com/.
[3] The KaZaA website, [Online]. Available: http://www.kazaa.com/.
[4] The Morpheus website, [Online]. Available: http://morpheus.com/.
[5] The Napster website, [Online]. Available: http://www.napster.com/.
[6] Yahoo website, [Online]. Available: http://www.yahoo.com/docs/info/faq.html.
[7] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley. "The Design and Implementation of an Intentional Naming System," in *Symp. Operating Syst. Principles*, pp. 186–201, 1999.
[8] R. Ahmed, R. Boutaba, F. Cuervo, Y. Iraqi, T. Li, N. Limam, J. Xiao, and J. Ziembicki. "Service naming in large-scale and multi-domain networks," *IEEE Commun. Surveys Tuts.*, vol. 7, no. 3, pp. 38–54, July 2005.
[9] R. Ahmed, N. Limam, J. Xiao, Y. Iraqi, and R. Boutaba. "Resource and service discovery in large-scale multi-domain networks," *IEEE Commun. Surveys Tuts.*, vol. 9, no. 4, pp. 2–30, Oct. 2007.
[10] S. Androutsellis-Theotokis and D. Spinellis. "A survey of peer-to-peer content distribution technologies," *ACM Comput. Surveys*, vol. 45, no. 2, pp. 195–205, Dec. 2004.
[11] G. Antoniou and F. V. Harmelen. Web Ontology Language: OWL. *Handbook on Ontologies in Information Systems*, pp 76–92, 2003.
[12] J. Aspnes, M. Safra, and Y. Yin. "Ranged hash functions and the price of churn," in *ACM-SIAM Symp. Discrete Algorithms*, pp. 1066–1075, Jan. 2008.
[13] J. Aspnes and G. Shah. "Skip graphs," in *Proc. Annual ACM-SIAM Symp. Discrete Algorithms (SODA)*, pp. 384–393, 2003.
[14] M. Balazinska, H. Balakrishnan, and D. Karger. "INS/Twine: A scalable peer-to-peer architecture for intentional resource discovery. in *Proc. International Conf. Pervasive Comput.*, pp. 195–210. Springer-Verlag, 2002.
[15] L. A. Barroso, J. Dean, and U. Holzle. Web search for a planet: The Google cluster architecture. *IEEE Micro*, vol. 23, no. 2, pp. 22–28, Apr. 2003.
[16] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer. "P2p content search: Give the web back to the people," in *International Workshop Peer-To-Peer Syst. (IPTPS)*, 2006.
[17] I. Bhattacharya, S. R. Kashyap, and S. Parthasarathy. "Similarity searching in peer-to-peer databases," in *Proc. IEEE Intl. Conf. Distributed Comput. Syst. (ICDCS)*, pp. 329–338, 2005.
[18] B. H. Bloom. "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, 1970.
[19] A. Bonifati, U. Matrangolo, A. Cuzzocrea, and M. Jain. "XPath lookup queries in P2P networks," in *Proc. ACM International Workshop Web Inf. Data Management (WIDM)*, pp. 48–55, New York, NY, USA, 2004. ACM Press.

[20] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard. *Web Service Architecture*, 2004. [Online]. Available. http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/.

[21] A. Broder and M. Mitzenmacher. "Network applications of bloom filters: A survey," *Internet Mathematics*, vol. 1, no. 4, pp. 485–509, 2003.

[22] D. Brookshier, D. Govoni, and N. Krishnan. *JXTA: Java P2P Programming*. SAMS, 2002.

[23] M. H. Burstein, J. R. Hobbs, O. Lassila, D. Martin, D. V. McDermott, S. A. McIlraith, S. Narayanan, M. Paolucci, T. R. Payne, and K. P. Sycara. "DAML-S: Web service description for the semantic web," in *Proc. International Semantic Web Conf. Semantic Web (ISWC)*, pp. 348–363, London, UK, 2002. Springer-Verlag.

[24] H. Cai and J. Wang. "Exploiting geographical and temporal locality to boost search efficiency in peer-to-peer systems," *IEEE Trans. Parallel Distributed Syst.*, vol. 17, no. 10, p. 1189–1203, Oct. 2006.

[25] M. Cai and M. Frank. "RDFPeers: A scalable distributed RDF repository based on a structured peer-to-peer network," in *International World Wide Web Conf. (WWW)*, 2004.

[26] D. Chamberlin, J. Siméon, S. Boag, D. Florescu, M. F. Fernández, and J. Robie. "XQuery 1.0: An XML query language," W3C recommendation, W3C, Jan. 2007. [Online]. Available: http://www.w3.org/TR/2007/REC-xquery-20070123/.

[27] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. "Making Gnutella-like P2P systems scalable," in *Proc. ACM SIGCOMM*, pp. 407–418, 2003.

[28] D. Choon-Hoong, S. Nutanong, and R. Buyya. *Peer-to-Peer Computing: Evolution of a Disruptive Technology*, ch. 2–Peer-to-Peer Networks for Content Sharing, pp. 28–65. Idea Group Inc., 2005.

[29] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. "Freenet: A distributed anonymous information storage and retrieval system," *Lecture Notes Comput. Science (LNCS)*, 2009, pp. 46–66, 2001.

[30] E. Cohen, A. Fiat, and H. Kaplan. "Associative search in peer-to-peer networks: Harnessing latent semantics. in *Proc. IEEE INFOCOM*, 2003.

[31] A. Crespo and H. Garcia-Molina. "Routing indices for peer-to-peer systems," in *Proc. International Conf. Distributed Comput. Syst. (ICDCS)*, 2002.

[32] S. E. Czerwinski, B. Y. Zhao, T. D. Hodes, A. D. Joseph, and R. H. Katz. "An architecture for a secure service discovery service," in *Proc. International Conf. Mobile Comput. Netw. (MOBICOM)*, pp. 24–35, 1999.

[33] S. Daswani and A. Fisk. "Gnutella udp extension for scalable searches (guess)," vol. 1.

[34] S. Decker, M. Schlosser, M. Sintek, and W. Nejdl. "Hypercup - hypercubes, ontologies and efficient search on P2P networks," in *International Workshop Agents Peer-to-Peer Comput.*, July 2002.

[35] E. Franconi, G. Kuper, A. Lopatenko, and I. Zaihrayeu. "The coDB robust Peer-to-Peer database system," in *Proc. Workshop Semantics Peer-to-Peer Grid Comput. International World Wide Web Conf. (WWW)*, May 2004.

[36] M. Fuchs, P. Wadler, J. Robie, and A. Brown. "XML schema: Formal description," W3C working draft, W3C, Sept. 2001. [Online]. Available: http://www.w3.org/TR/2001/WD-xmlschema-formal-20010925/.

[37] L. Galanis, Y. Wang, S. Jeffery, and D. DeWitt. "Locating data sources in large distributed systems," in *Proc. VLDB Conf.*, 2003.

[38] P. Ganesan, Q. Sun, and H. Garcia-Molina. "Adlib: A self-tuning index for dynamic peer-to-peer systems," in *Proc. International Conf. Data Eng. (ICDE)*, pp. 256–257, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

[39] J. Garofalakis, Y. Panagis, E. Sakkopoulos, and A. Tsakalidis. "Web service discovery mechanisms: Looking for a needle in a haystack?" in *International Workshop Web Eng.*, 2004.

[40] T. Gu, D. Zhang, and H. K. Pung. "A two-tier semantic overlay network for p2p search," in *Proc. Intl. Conf. Parallel Distributed Syst.*, Dec. 2007.

[41] E. Guttman, C. Perkins, J. Veizades, and M. Day. "Service Location Protocol (SLP), version 2," Technical report, IETF, RFC2608, [online]. Available: http://www.ietf.org/rfc/rfc2608.txt, June 1999.

[42] M. Harren, J. M. Hellerstein, R. Huebsch, B. T. Loo, S. Shenker, and I. Stoica. "Complex queries in DHT-based peer-to-peer networks," in *Proc. International Workshop Peer-to-Peer Syst. (IPTPS)*, pp. 242–259, 2002.

[43] N. Harvey, M. B. Jones, S. Saroiu, M. Theimer, and A. Wolman. "SkipNet: A scalable overlay network with practical locality properties," in *Proc. USENIX Symp. Internet Technol. Syst. (USITS)*, Mar. 2003.

[44] S. Herschel and R. Heese. Humboldt Discoverer: A semantic P2P index for PDMS. in *Proc. International Workshop Data Integration Semantic Web (DISWeb'05)*, June 2005.

[45] T. Howes. "Rfc 2254: The string representation of ldap search filters," 1997.

[46] H. Hu and A. Seneviratne. "Autonomic peer-to-peer service directory," *IEICE/IEEE Joint Special Section Autonomous Decentralized Syst.*, vol. E88-D, no. 12, pp. 2630–2639, Dec. 2005.

[47] T. H. Hu, S. Ardon, and A. Seneviratne. Semantic-laden peer-to-peer service directory. in *Proc. IEEE Intl. Conf. P2P Comput.*, pp. 184, 2004.

[48] X. Jin, W. P. Ken Yiu, and S. H. Gary-Chan. "Supporting multiple-keyword search in a hybrid structured peer-to-peer network," in *Proc. IEEE International Conf. Commun. (ICC)*, pp. 42–47, Istanbul, June 2006.

[49] Y. Joung, L. Yang, and C. Fang. "Keyword search in DHT-based peer-to-peer networks," *IEEE J. Sel. Areas Commun. (JSAC)*, vol. 25, no. 1, pp. 46–61, Jan. 2007.

[50] Y. Kalfoglou and M. Schorlemmer. "Ontology mapping: The state of the art," *Knowledge Eng. Rev. J. (KER)*, vol. 18, no 1, pp. 1–31, 2003.

[51] V. Kalogeraki, D. Gunopulos, and D. Zeinalipour-Yazti. "A local search mechanism for peer-to-peer networks," in *Conf. Inf. Knowledge Management (CIKM)*, 2002.

[52] D. R. Karger and M. Ruhl. "Simple efficient load balancing algorithms for peer-to-peer systems," *Theory Comput. Syst.*, vol. 39, vol. 6, pp. 787–804, Nov. 2006.

[53] F. B. Kashani, C. C. Chen, and C. Shahabi. "Wspds: Web services peer-to-peer discovery service," in *Proc. Intl. Conf. Internet Comput.*, pp. 733–743, 2004.

[54] M. Kay, M. F. Fernández, S. Boag, D. Chamberlin, A. Berglund, J. Siméon, and J. Robie. "XML path language (XPath) 2.0. W3C recommendation, W3C," Jan. 2007. [Online]. Available: http://www.w3.org/TR/2007/REC-xpath20-20070123/.

[55] J. Kim and G. Fox. "A hybrid keyword search across peer-to-peer federated databases," in *Proc. East-European Conf. Advances Databases Inf. Syst. (ADBIS)*, Sept. 2004.

[56] J. Kleinberg. "The small-world phenomenon: an algorithm perspective," in *Proc. ACM Symp. Theory Comput.*, 2000.

[57] G. Koloniari and E. Pitoura. "Peer-to-peer management of xml data: Issues and research challenges," *ACM SIGMOD Record*, vol. 34, no. 2, pp. 6–17, June 2005.

[58] G. Koloniari and E. Pitoura. "Peer-to-peer management of XML data: Issues and research challenges," *ACM SIGMOD Record*, vol. 34, no. 2, pp. 6–17, 2005.

[59] A. Kumar, J. Xu, and E.W. Zegura. "Efficient and scalable query routing for unstructured peer-to-peer networks," in *Proc. IEEE INFOCOM*, pp. 1162–1173, 2005.

[60] O. Lassila and R. R. Swick. "Resource description framework (RDF) model and syntax specification, supersed work, W3C," Feb. 1999. [Online]. Available: http://www.w3.org/TR/1999/REC-rdf-syntax-19990222.

[61] C. Lee and S. Helal. "Protocols for service discovery in dynamic and mobile networks," *International J. Comput. Research*, vol. 11, no. 1, pp. 1–12, 2002.

[62] M. Li, W. Lee, and A. Sivasubramaniam. "Neighborhood signatures for searching P2P networks," in *Proc. Seventh International Database Eng. Appl. Symp. (IDEAS)*, pp. 149–159, 2003.

[63] Y. Li, F. Zou, Z. Wu, and F. Ma. "PWSD: A scalable web service discovery architecture based on peer-to-peer overlay network," in *Proc. APWeb, Lecture Notes Ccomput. Science (LNCS)*, vol. 3007, 2004.

[64] L. Liu, K. D. Ryu, and K. Lee. "Supporting efficient keyword-based file search in peer-to-peer file sharing systems," in *Proc. GLOBECOM*, 2004.

[65] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. "A survey and comparison of peer-to-peer overlay network schemes," *IEEE Commun, Surveys*, pp. 72–93, Second Quarter 2005.

[66] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. "A survey and comparison of peer-to-peer overlay network schemes," *IEEE Commun. Surveys Tuts.*, vol. 7, no. 2, pp. 72–93, 2005.

[67] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. "Search and replication in unstructured peer-to-peer networks," in *Proc. International Conf. Supercomput. (ICS)*, 2002.

[68] P. Maymounkov and D. Mazireres. "Kademlia: A peer-to-peer information system based on the XOR metric," in *Proc. International Workshop Peer-to-Peer Syst. (IPTPS)*, pp. 53–65. Springer-Verlag, Mar. 2002.

[69] B. A. Miller, T. Nixon, C. Tai, and M. D. Wood. "Home networking with Universal plug and play," *IEEE Commun. Mag.*, pp. 104–109, Dec. 2001.

[70] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu. "Peer-to-peer computing," Tech. Rep. HPL-2002-57R1, HP Labs, 2002.
[71] M. Montebello and C. Abela. "DAML enabled web service and agents in semantic web," in *Workshop Web, Web Services Database Syst., Lecture Notes Ccomput. Science (LNCS)*, 2003.
[72] W. Nejdl, M. Wolpers, W. Siberski, C. Schmitz, M. Schlosser, I. Brunkhorst, and A. Loser. "Super-peer-based routing strategies for RDF-based peer-to-peer networks," *J. Web Semantics*, vol. 1, no. 2, pp. 177–186, Feb. 2004.
[73] W. Siong Ng, B. Chin Ooi, K. Lee Tan, and A. Zhou. "PeerDB: A P2P-based system for distributed data sharing," in *Proc. International Conf. Data Eng. (ICDE)*, pp. 633–644, 2003.
[74] M. Tamer Ozsu and P. Valduriez. *Principles of Distributed Database Systems*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1991.
[75] E. Prud'Hommeaux and A. Seaborne. "SPARQL query language for RDF," Working Draft WD-rdf-sparql-query-20061004, *World Wide Web Consortium (W3C)*, Oct. 2006.
[76] W. Pugh. "Skip lists: A probabilistic alternative to balanced trees," *Commun. ACM*, vol. 33, no. 6, pp 668–676, 1990.
[77] M. Rabin. "Fingerprinting by random polynomials," Technical report, CRCT TR-15-81, Harvard University, 1981.
[78] E. Rahm and P. Bernstein. "A survey of approaches to automatic schema matching," *International J. Very Large Data Bases (VLDB)*, vol. 10, no. 4, pp. 334–350, 2001.
[79] Raghu Ramakrishnan and Johannes Gehrke. *Database Management Systems*. McGraw-Hill Professional, 2002.
[80] R. Ranjan, A. Harwood, and R. Buyya. "Peer-to-peer-based resource discovery in global grids: A tutorial," *IEEE Commun. Surveys Tuts.*, vol. 10, no. 2, pp. 6–33, 2008.
[81] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. "A scalable content-addressable network," in *Proc. ACM SIGCOMM*, pp. 161–172, 2001.
[82] S. Rhea and J. Kubiatowicz. "Probabilistic location and routing," in *Proc. IEEE INFOCOM*, 2002.
[83] J. Risson and T. Moors. "Survey of research towards robust peer-to-peer networks: Search methods," *Comput. Netw.*, vol. 50, no. 17, pp. 3485–3521, Dec. 2006.
[84] P. Rompothong and T. Senivongse. "A query federation of UDDI registries," In *Proc. International Symp. Inf. Commun. Technol. (ISICT)*, pp. 578–583, 2003.
[85] P. Rosch, K. Sattler, C. Weth, and E. Buchmann. "Best effort query processing in dht-based p2p systems," in *Proc. Intl. Conf. Data Eng. (ICDE)*, 2005.
[86] K. W. Ross and D. Rubenstein. Tutorial on p2p systems. presented at Infocom, 2004.
[87] A. Rowstron and P. Druschel. "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. in *Proc. IFIP/ACM International Conf. Distributed Syst. Platforms (Middleware)*, Heidelberg, Germany, Nov. 2001.
[88] H. Sagan. *Space-filling Curves*. Springer-Verlag, 1994.
[89] emphSalutation Consortium. Salutation architecture specification version 2.0c, June 1999.
[90] M. Schlosser, M. Sintek, S. Decker, and W. Nejdl. "A scalable and ontology-based P2P infrastructure for semantic web services," in *Proc. International Conf. peer-to-peer Comput. (P2P)*, Sept. 2002.
[91] C. Schmidt and M. Parashar. "Enabling flexible queries with guarantees in P2P systems," *IEEE Internet Comput.*, vol. 8, no. 3, pp. 19–26, June 2004.
[92] C. Schmidt and M. Parashar. "Peer-to-peer approach to web service discovery. in *WWW: Internet Web Inf. Syst.*, vol. 7, pp. 211–229, 2004.
[93] A. Seaborne. "RDQL - a query language for RDF," (member submission). Technical report, W3C, Jan. 2004.
[94] S. Shi, G. Yang, D. Wang, J. Yu, S. Qu, and M. Chen. "Making peer-to-peer keyword searching feasible using multi-level partitioning," in *Proc. International Workshop Peer-to-Peer Syst. (IPTPS)*, pp. 151–161. Springer, 2004.
[95] P. Shvaiko and J. Euzenat. "A survey of schema-based matching approaches," *J. Data Semantics*, vol. IV, pp. 146–171, 2005.
[96] C. M. Sperberg-McQueen, Tim Bray, Eve Maler, Jean Paoli, and François Yergeau. "Extensible markup language (XML)," 1.0 (fourth edition). W3C recommendation, W3C, Aug. 2006. [Online]. Available: http://www.w3.org/TR/2006/REC-xml-20060816.
[97] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. "Chord: A scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Trans. Netw. (TON)*, vol. 1, pp. 17–32, 2003.
[98] M. Stokes. "Gnutella2 specifications part one," [Online]. Available: http://www.gnutella2.com/gnutella2 search.htm.
[99] Sun Microsystems. *Jini Technology Core Platform Specification*, Oct. 2000. [Online]. Available: http://www.sun.com/jini/specs/.
[100] C. Tang and S. Dwarkadas. "Hybrid global-local indexing for efficient Peer-to-Peer information retrieval," in *Proc. Symp. Netw. Syst. Design Implementation (NSDI)*, June 2004.
[101] C. Tang, Z. Xu, and M. Mahalingam. "PSearch: Information retrieval in structured overlays," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp 89–94, 2003.
[102] D. Tsoumakos and N. Roussopoulos. "Adaptive probabilistic search for peer-to-peer networks," in *Proc. International Conf. Peer-to-Peer Comput. (P2P)*, 2003.
[103] D. Tsoumakos and N. Roussopoulos. "Analysis and comparison of p2p search methods," in *Proc. International Conf. Scalable Inf. Syst.*, pp. 25–39, New York, NY, USA, 2006. ACM Press.
[104] UDDI Consortium. UDDI Technical White Paper, 2002. [Online]. Available: http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf.
[105] M. Uschold and M. Gruninger. "Ontologies: Principles, methods and applications," *Knowledge Sharing Rev.*, vol. 11, no. 2, 1996.
[106] H. Wache, T. Voegele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Huebner. "Ontology-based integration of information - a survey of existing approaches," in *Proc. Workshop Ontologies Inf. Sharing International Joint Conf. Artificial Intelligence (IJCAI)*, pp. 108–117, 2001.
[107] B. Yang and H. Garcia-Molina. "Improving search in peer-to-peer networks," in *Proc. International Conf. Distributed Comput. Syst. (ICDCS)*, 2002.
[108] B. Zhao, L. Huang, J. Stribling, S. Rhea, A. Joseph, and J. Kubiatowicz. "Tapestry: A resilient global-scale overlay for service deployment," *IEEE J. Sel. Areas Commun. (JSAC)*, vol. 22, no. 1, pp. 41–53, Jan. 2004.
[109] F. Zhu, M. Mutka, and L. Ni. "Classification of service discovery in pervasive computing environments," Technical Report MSU-CSE-02-24, Michigan State University, East Lansing, 2002.

Reaz Ahmed is working as Assistant Professor at the department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh. He received the PhD. Degree in Computer Science from the University of Waterloo, in 2007. He received the MSc. and BSc. degrees in Computer Science from BUET in 2002 and 2000, respectively. He received the IEEE Fred W. Ellersick award 2008. His research interests include wide area service discovery, loosely-coupled distributed databases and content-sharing peer-to-peer networks with focus on search flexibility, efficiency and robustness.

Raouf Boutaba is a Professor of Computer Science and a Cheriton Faculty Fellow at the University of Waterloo (Canada). His main research interests are in network, resource and service management. He is the founding Editor-in-Chief of the IEEE Transactions on Network and Service Management and on the editorial boards of other journals. He served as a distinguished lecturer of the IEEE Communications and the IEEE Computer Societies. He also served as the chairman of the IEEE Technical Committee on Information Infrastructure and the IFIP Working Group on Network and Distributed Systems Management. He has received several recognitions such as the Premiers research excellence award, the IEEE Harold Sobol, Fred W. Ellersick, Joe LoCicero awards and the Don Stokesburry award.