Semantic Data Management in Peer-to-Peer E-Commerce Applications

Yosi Ben-Asher and Shlomo Berkovsky*

Computer Science Department, University of Haifa
31905 Haifa, Israel
{yosi, slavax}@cs.haifa.ac.il

Abstract. Weakly organized structure of Peer-to-Peer systems may cause severe data management problems and high communication overheads. We partially resolve these problems by developing a novel semantic approach to efficiently create, search and organize data objects in E-Commerce Peer-to-Peer applications. The approach is based on the notion of Unspecified Ontology (UNSO). Unlike many existing systems using a global predefined ontology, UNSO approach assumes that the ontology is not fully defined, leaving some parts of it to be dynamically specified by the users. The data objects inserted to the system organize a multi-layered hypercube graph topology, providing a stable infrastructure for efficient semantic search and routing operations. The proposed method has a potential of becoming a practical infrastructure for Peer-to-Peer data management applications.

Keywords: Peer-to-Peer Systems, Data Management, Ontology, E-Commerce.

1 Introduction

Peer-to-Peer (P2P) [21] technology offers a solid alternative to the traditional Client-Server model of computing. While Client-Server model typically bases on a single or small number of servers, in P2P systems every user (peer) acts as both the client and the server at the same time, and provides a portion of the system capability. Thus, P2P technology allows a dynamic set of users to efficiently share resources without any centralized management. The shared resources are computing power (e.g., in distributed computation), data (e.g., in large-scale file sharing), bandwidth (e.g., data transfer from multiple sources) and others. As a result, the advantages of P2P technology over the Client-Server model include roughly unlimited scalability, high privacy and anonymity of the users, and low costs. Sharing and aggregation of the resources guarantees robustness and high availability of P2P systems.

In this work we examine the issue of developing a P2P infrastructure supporting a dynamic semantic management (i.e., insertion and search) of general-purpose E-Commerce advertisements (in short, *ads*). The infrastructure is capable of managing E-Commerce ads of both *supply* and *demand* types. Supply ads are ads, where the users offer a product or a service in exchange for a payment, whereas in demand ads

* This work was supported by a research grant from Caesarea Edmond Benjamin de Rothschild. Foundation Institute for Interdisciplinary Applications of Computer Science (CRI).

S. Spaccapietra et al. (Eds.): Journal on Data Semantics VI, LNCS 4090, pp. 115 – 142, 2006.

the users seek for a product or for a service provided by other users. The main functionality of the proposed system is to identify matching between appropriate demand and supply ads. This is further referred as publish-locate functionality. Note that we do not aim at performing higher functions, such as selecting the best offer, conducting public auctions or implementing market clearing (see [9] and [20] for a discussion on distributed systems performing these tasks).

In the state-of-the-art E-Commerce systems, a user publishing or searching an ad, is usually required to fill-in a predefined form describing the matter of the ad. For example, CarSmart (*http://www.carsmart.com*) users searching for a car are asked to fill-in a form containing the following fields: manufacturer name, geographical location, and price range. On the contrary, in another cars site, Motocar (*http://www.motocar.co.il*), the users are asked to fill-in a more complicated form, containing the following fields: producer name, model, range of production years, gearbox type, engine volume and number of previous owners. Although both sites refer to the same type of products, the forms (and even the names or parallel fields in the form) are different, such that a search query launched by the user in one site, is incompatible to the other.

This approach, exploiting predefined forms containing a set of attributes describing the objects, is further referred as ontology-based approach. According to [13], ontology is a formal explicit specification of a domain. Thus, the set of attributes, describing the objects from a particular domain, is considered as the ontology of a domain, whereas the attributes are the slots of the ontology.

HyperCuP [32] proposed a flexible ontology-based hypercube topology for P2P data management. It used a single predefined ontology to classify users as providers of particular information associated with the ontology slots. This classification determined the position of the user in the underlying hypercube and allowed location of any desired information in a bounded number of steps through a semantic routing (see [27] and [5] for a discussions on semantic routing in P2P systems). Thus, HyperCup formed an alternative to distributed hashing [29], [33], [26], network flooding [3], local routing tables [10], gossiping [2], and others fundamental search techniques in P2P systems. Additionally, HyperCuP proposed decentralized algorithms, capable of constructing and maintaining highly-connected hypercube graph, stable to dynamic joins and departures of users.

However, HyperCuP approach, requiring a single predefined ontology, is applicable only to a limited set of domains and is inappropriate for general-purpose E-Commerce systems, implying dynamic and a-priori unknown set of objects. A possible solution might be allowing users to add new types of ontological forms. However, this will flood the system with multiple (partially similar and overlapping) ontologies. As a negative example, consider a setting, where a user publishing an object and a user looking for the same object, are using slightly different ontologies. Another solution might be developing a global comprehensive ontology, comprising as many domain ontologies as possible. However, projecting this ontology on the underlying hypercube will result in a huge, sparse and barely manageable structure. Moreover, sharing of the single ontology by all the users will obstruct it from being expanded.

All these restrictions contradict the decentralized spirit of P2P networks and raise an issue of developing a flexible mechanism for managing a dynamic set of

ontologies. Recent researches, e.g., [23] and [34], aim at resolving the ontologies management issue through integrating local ontologies used by various systems. These works try to overcome the autonomous nature of P2P systems and the resulted heterogeneity of the ontologies by developing semantic mappings between the ontologies. The mappings exploit semantic knowledge about the ontology slots for the purposes of identifying commonalities between the ontologies and matching different terms describing similar underlying data.

Conversely, in this work we developed a novel approach of an *UNSpecified Ontology* (UNSO) for the management of E-Commerce ads. Instead of trying to identify the matching between the ontologies, UNSO presumes that the ontology is not fully defined, and parts of it can be dynamically specified by the users. This broader notion of ontology allows UNSO to be exploited in different application domains, rather than just in the domains defined a-priori by the ontology experts.

To maintain the semantic routing, UNSO extends the original hypercube graph of HyperCuP to a *Multi-Layered Hypercube* (MLH). MLH can be schematically depicted as a hypercube where each node is recursively constructed of another hypercube. Hashing mechanism is used to deal with the unspecified nature of the ontology, and with the variety of terms mentioned in the ads. Besides that, hashing uniformly distributes the ads among the MLH to guarantee equivalent load partitioning.

The generated structure facilitates the semantic routing, similar to the routing of HyperCuP. To eliminate ambiguity and enhance the precision, the terms mentioned in the ads undergo simple semantic standardization using WordNet [11]. In summary, UNSO expands the traditional notion of ontological data management and provides a novel technique for a decentralized management of a dynamic set of ontology-based ads, while keeping the essential publish-locate functionality. The fact that the users are not forced to share any predefined ontology simplifies the use of the system by inexperienced users and increases the chance of successful searches.

The rest of the paper is organized as follows. In section 2 we review the research efforts in P2P computing, the major classes of P2P applications, their advantages and shortcomings. Section 3 discusses the works in the area of ontology-based management in P2P systems and particularly HyperCuP [32]. Section 4 presents the notion of unspecified ontology, and discusses the generalization of a fixed ontology to UNSO. Section 5 discusses the details of UNSO implementation. In section 6 we present the experimental results, showing the performance of UNSO. Finally, section 7 concludes the work and discusses the directions of further research.

2 Peer-to-Peer Data Management

P2P computing refers to a subclass of distributed computing, where the system functionality is achieved in a decentralized way by unifying a set of distributed resources, such as computing power, data and network traffic. P2P systems usually lack a designated centralized management, rather depending on the voluntary contribution of resources by the users. These systems are usually characterized by one or more of the following advantages: cost sharing/reduction, improved scalability/reliability, resource aggregation and operability, increased autonomy, dynamism, anonymity/privacy and ad-hoc communication and collaboration [21].

The first generation of P2P systems was based on three classical architectures: mediated P2P architecture (e.g., [30]), pure P2P architecture (e.g., [10] and [26]), and hybrid architecture (e.g., [38] and [22]). An elaborate comparison, advantages and shortcomings of the above architectures, and discussion on the typical applications of each architecture can be found in [6].

Basically, they all were designed for a large-scale data sharing. Applications, such as Napster [30], Freenet [10] and Gnutella [3], allowed users to download data (mainly multimedia files), shared by other users. Performance of these systems suffered from severe problems. For example, in Napster a cluster of central servers, called super-peers maintained the indices of the files shared by the users. Flooding search algorithm of Gnutella limited the scalability of the system and did not allow proper functioning over a heterogeneous set of users. Freenet, despite being fully decentralized and employing efficient routing algorithms, could not guarantee reliable data location. This led to a development of content-addressable P2P systems.

A number of similar fully decentralized content-addressable P2P systems, such as CAN [26], Pastry [29], Chord [33] and some others, are referred as the second generation of P2P systems. They implement highly scalable self-organizing infrastructure for a fault-tolerant routing over distributed hashing data management mechanism (DHT). In these systems, the users and the data objects are assigned unique identifiers (respectively, *user-ids* and *keys*) from a sparse space. Data objects are inserted and located through *put(key,user-id)* and *get(key)* primitives in a bounded number of routing network hops. An elaborate description of and survey of P2P content-addressable distribution technologies can be found in [4].

In Pastry [29], *user-ids* and keys are *128* bit vectors. Routing tables of the connected users contain *O(logN)* rows and *16* columns, where *N* is the total number of users. The entries in the row *n* of the routing tables refer to the logical neighbors, whose *user-ids* share the first *n* digits with the *user-id* of the current users, whereas the digit *(n+1)* of the *user-id* in column *m* of the row *n* is *m*. Unlike Pastry, Chord [33] uses a one-dimensional circular *160*-bit space and instead of the prefix-based neighbors table, Chord users maintain a finger table, containing *user-ids* and addresses of the other connected users. Entry number *i* in the table of a user *n* refers to another user with the smallest *user-id*, clockwise from $n+2^{i-1}$. The first entry in the table points to the direct successor of user *n*, whereas the following entries point to the users at repeatedly doubling distances from *n*. CAN [26] is operated over *d*-dimensional toroidal space, where each user is associated with a hypercubal zone of the space, and its neighbors are the other users managing the adjacent hypercube zones.

The routing algorithm of content-addressable systems is based on Plaxton routing algorithm, developed in [25]. Plaxton algorithm was not designed for P2P systems, rather for graphs with a static nodes' population. The main idea of Plaxton algorithm is correcting a single digit of address in every routing step. Consider the following example. User *1234* receives a message, addressed to user *1278* (note that the first two digits of the address already match). The message is forwarded to user *1275* (since there the first three digits will match). To support this routing, each user maintains a data structure of logical neighbors that match *i*-digits length prefix of its own *user-id*, but differ in the *(i+1)th* digit. To maintain a connected system with *N* users, each user is connected to *O(logN)* neighbors. Since a single digit of address is

corrected each time the message is routed, the total length of the routing path is *O(logN)* hops.

In P2P implementations of Plaxton routing algorithm, the routed message is continuously forwarded to the user, whose *user-id* is closer to the *user-id* of the addressee than the current user. Although each one of the above DHT-based systems (Pastry, CAN, and Chord) employs slightly different variant of Plaxton routing algorithm, they all outperform the routing algorithms used in the first generation of P2P systems. Their communication overhead is significantly lower due to the fact that the messages are routed to the relevant users only.

However, DHT-based system rely on the hashing primitives of *put(key,user-id)* and *get(key)*. Thus, one of their major limitations is their support in exact-match searches only. For example, consider two similar, but not identical keys: key_1 and key_2. The results of their insertions into the system through hashing-based *put* primitive will usually be absolutely different. Therefore, only the searches, specifying the exact *key* that was used when the *key* was inserted, will succeed to find it, and an approximated search can not be operated.

Possible solutions for this issue are discussed in [15]. The paper compares various approaches facilitating complex range queries in DHT-based P2P networks. An architecture for relational data sharing in Chord-based systems was proposed in [14]. It exploits hashing mechanisms on the possible values within a given range to facilitate answering range queries. Solution for range queries problem in CAN-based systems are proposed in [30]. In this work, CAN space is partitioned to zones, in a way that allows the peers managing the zones to store information regarding the range of values that are mapped to this zone, as well as to the neighbor zones. This facilitates resolving range queries through their iterative forwarding to the zone that is managing the requested range of values.

Basically, the above approaches focus on constructing search indices enabling to perform database-like queries using operators such as selection, join, aggregation and others. These indices can help only in domains where the terminology of the *keys* names is agreed between the users, e.g., in music files, where the songs names are known. However, in domains with no well-defined naming standards, particularly in general-purpose E-Commerce applications, different users might use different terms to describe the same (or highly similar) object. Thus, in order to develop a generic infrastructure facilitating matching of similar E-Commerce ads, the ads should be preprocessed to a semi-structured form of attributes (slots) and their respective values. To achieve that, there is an emergent need for more complex type of P2P systems, built upon users that explicitly use semantic structures to describe the matters of their ads. This approach is further referred as ontology-based approach.

3 Ontology-Based Data Management

One of the basic concepts in the semantic data management is *ontology* (according to the Wikipedia, it is “the study of existence and the basic categories thereof”). Practically, it provides both human-understandable and machine-processable mechanisms, allowing enterprises and application systems to collaborate in a smart way. According to [13], ontology is a formal shared conceptualization of a particular domain. It is

used to describe structurally heterogeneous and distributed information sources, such as those found on the Web, and acts as a standardized reference model, providing a stable baseline for shared understanding of the domain knowledge.

Ontological metadata facilitates a standardized access to the domain knowledge. Existing approaches of ontology-based data management assume a setting where the data sources share a single ontology allowing the access to the data. This technique of shared ontology was implemented in HyperCuP [32] that proposed a set of dynamic P2P algorithms for maintenance of ontology-based hypercube-like graph topology supporting semantic search and broadcast operations.

HyperCup developed a flexible ontology-based P2P platform generating a hypercube-like graph of users, where each user is treated as a data provider. HyperCup needs predefined domain ontology to be exploited, such that the dimensions of the hypercube will match the ontology slots, i.e., a set of attributes characterizing the domain objects. According to the above ontology, each user is categorized as a provider of particular data. This categorization determines the location of the user within the hypercube. Thus, the hypercube is virtually constructed of the connected users, whereas each user maintains a data structure of its respective neighbors. For example, in *3*-dimensional hypercube, a node located in coordinates (*x,y,z*) will be connected to *6* logical neighbors: (*x+1,y,z*), (*x-1,y, z*), (*x,y+1,z*), (*x,y-1,z*), (*x,y,z+1*) and (*x,y,z-1*). The user providing multiple data objects from the same domain, or data objects from multiple domains, will maintain a set of hypercube locations, such that a separate data structure of neighbors will reflect each location.

The hypercube-like P2P structure was chosen due to its logarithmic diameter, increased fault tolerance and the symmetry that guarantees equal load of the users. The hypercube dimension d and the range of possible values in the dimensions (further referred as the *coordinates range*) k determine the maximal number of users connected to the hypercube. A complete hypercube contains at most $N_{max}=k^d$ users, where every user is connected to two logical neighbors in each dimension, resulting in $N_n=2d$ neighbors. This topology generates a symmetric structure where the load of the connected users in the system is similar, as each user holds roughly equal functionality in terms of routing load.

Any edge in the hypercube, connecting a pair of adjacent users X and Y, is assigned a numeric value, referred as *rank*. When user Y is a neighbor of user X over a dimension i, the rank of the edge, connecting X and Y, is i. Thus, the edges rank ranges from *0* to *d-1*. Any user T in the hypercube can act as an initiator of search or broadcast operation, which is performed as follows. The message, jointly with the rank of the connecting edge, is sent to all the neighbors of the initiating user T. Upon receiving a message, other users forward it only over the edges, whose ranks are higher than the rank of the edge the message was received from. This guarantees that each user in the graph will receive the message exactly once, and also that any connected user will be reached in $O(d)$ routing hops.

Consider the following example hypercube with a dimension $d=3$ and the coordinates range $k=2$ (figure 1a). Eight users, numbered from *1* to *8*, are connected and form a complete hypercube. Each user is connected to exactly $d=3$ logical neighbors, and the ranks of the connecting edges are *0*, *1*, or *2*. For example, in respect to user *8*, user *1* is regarded as *0*-neghbor, user *3* is *1*-neighbor, and user *4* is *2*-neighbor. Let

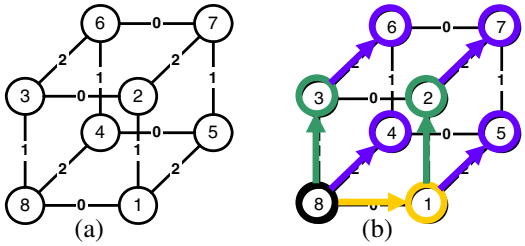

Fig. 1. HyperCuP Example (a) 3-dimensional HyperCuP structure with *8* connected users; (b) Broadcast procedure stages over the example HyperCuP structure

user *8* be the initiator of the broadcast operation. The messages are sent to the neighbors, i.e., users *1*, *3* and *4*. Upon receiving a message over *0*-rank edge, user *1* forwards it to *1*- and *2*-neghbors, i.e., users *2* and *5*. User *3* forwards it to *2*-neighbor, user *6*, and this broadcast through "higher-rank forwards" continues until all the nodes in the hypercube are covered (figure 1b). Obviously, no node receives the broadcasted message more than once, and the longest path in the hypercube is *d=3* hops long.

HyperCuP also proposes a dynamic P2P algorithm for hypercube construction and maintenance. The algorithm is based on the idea that a user can manage not only a single node, but also a number of nodes in the hypercube graph. This is required in order to *simulate* the missing users in the topology of the next complete hypercube, which is implicitly maintained in any topology state. For example, consider node *4* simulating three missing nodes of the hypercube (figure 2a). The simulated nodes are schematically illustrated by the dashed edges *1-4*, *2-4* and *3-4*, as node *4* acts as a logical neighbor of nodes *1*, *2* and *3*.

When a new user connects the network, he takes his place (according to the data provided) in the next complete hypercube, releases the user that previously managed that node and starts functioning as a real hypercube node. For example, if a new user, which should be positioned in node *5*, is connected, he is routed to one of the existing logical neighbors of node *5*, i.e., either to the user maintaining node *1* or to the user maintaining node *4*. As node *5* is practically simulated by the user maintaining node *4*, the new user contacts the user of node *4*, builds a real edge between them and takes

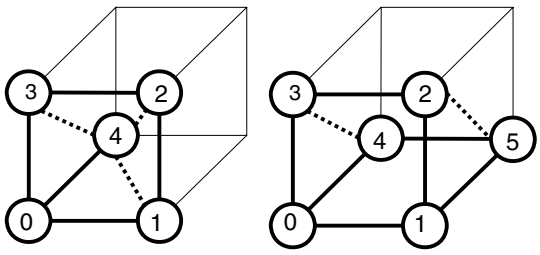

Fig. 2. (a) Implicitly preserved topology of the next complete hypercube; (b) Join of a new node

part of its functionality, i.e., builds a real edge with node *1* and starts simulating the neighbor of node *2* (fig. 2b).

When a user disconnects, one of the remaining logical neighbors takes the responsibility for the node, previously managed by the leaving user. Since the next complete hypercube is constantly maintained, previously discussed broadcast and search operations are not affected by sporadic joins and departures of users. An elaborate description of the way the HyperCuP topology is maintained, and examples of such departures and joins, can be found in [32].

As already stated, users are classified as providers of particular contents. A single predefined ontology defining the domain semantics, inherently organizes the users providing the same or similar contents, in *concept clusters* using the above construction and maintenance algorithm. Note that the similarity of users within a cluster is significantly higher that the similarity of arbitrarily chosen users. This facilitates querying the generated topology and efficiently routing the queries only to the clusters (and users) that can potentially answer it.

For example, consider the following simple ontology of cars domain, used to construct a *3*-dimensional HyperCuP (figure 1a): dimension *0* distinguishes between manual(*0*) and automatic(*1*) gearbox, dimension *1* stands for USA(*0*) or non-USA(*1*) produced cars, and dimension *2* for metallic(*0*) or non-metallic (*0*) color of the car. Clearly, a query for automatic cars produced in the USA is routed to nodes *1* and *5*, as only these nodes store the requested type of cars.

However, the predefined ontology constitutes one of the main drawbacks of HyperCuP approach. If used in a general-purpose E-Commerce application capable of supporting various types of transactions and objects, HyperCuP would require global all-inclusive ontology to be exploited. This is a severe limitation, as the state-of-the-art ontologies are usually limited to a single application domain, and even ontologies from the same domain can be highly heterogeneous due to existence of different *views* on the domain data.

Earlier research efforts, such as [16] and [12], focus on the issue of merging ontologies and generating a global ontology. To create a unified view enabling to integrate data originated by different ontologies, their semantic reconciliation is required. Merging two ontologies means creating a new ontology comprising the slots of both ontologies. Merging process is mainly based on recognizing relationships and commonalities between the slots of the ontologies. The ontology, generated as a result of merging, matches both merged ontologies.

However, the issue of constructing all-inclusive ontology is a controversial issue (see [35] for an elaborate discussion). Besides the philosophical question of creating a *universal* ontology, it leads to a setting where updates of the ontology are conducted through a central point of management, unacceptable in pure P2P networks. Moreover, existence of this global ontology forces all the users to use it, contradicting the decentralized free spirit of P2P networks.

In P2P, we rather face a situation, where individual users maintain their own views of the domain ontology. However, since data sharing is one of the primary motivations behind the state-of-the-art P2P systems, we need to find a way to distributively manage multiple dynamic ontologies escribing the domain data objects. The task of sharing heterogeneous ontological data became an important research direction in P2P community. It is referred in the literature as the *Data Integration* problem [8].

Traditional data integration approaches adapted from the distributed databases community [36], assume a central unified representation of the data objects, which contradicts the decentralized nature of P2P systems and suffers from scalability problem. In addition to the lack of central point of management, P2P data integration techniques have to cope with multiple (sometimes ambiguous and overlapping) and highly dynamic ontologies representing the data. Most of the techniques achieve the goal of sharing and integrating data through developing semantic mapping mechanisms, describing the relationships between the slots of the ontologies [19].

Automatic approaches for ontology (or schema) mapping in P2P systems do not require central ontology, but perform the mapping at one of three levels:

- Pair mapping – the mappings are performed between the relevant peers only [7]. In this setting, one of the peers involved in the mapping process, defines specific translation and coordination rules allowing to relate the slots of his ontology to the ontology of the other peer.
- Peer-mediated mapping – a generalization of pair mapping, where one of the peers defines a mapping that relates to a number of peers. This approach was implemented in [34] through machine learning technique that exploited various evidences of semantic similarity of the ontology slots. In [23], the authors used Information Retrieval techniques for extracting descriptive keywords, which were exploited later for identifying appropriate slots in the ontologies. The mappings are generated 'on-the-fly', whenever they are needed for the integration of data.
- Super-peer mediating mapping – semi-centralized approach, where the mappings are performed at the super-peers level. In this setting, super-peers is responsible for managing the mappings for the ontologies of the underlying peers, whereas super-peer to super-peer mappings facilitate data sharing between any peers. This approach was implemented in Edutella [22], RDF-based P2P platform for educational data sharing.

Most of the state-of-the-art P2P systems refer to the semantic heterogeneity between the underlying ontologies. However, they assume that the mappings between the ontologies can be constructed. Although we do not negate the possibility of constructing such mappings, we highlight an observation that the process of creating these mappings might be one of the most challenging tasks in a heterogeneous and highly dynamic P2P realm. Alternatively, in this work we propose another way to overcome the issue of semantic heterogeneity in P2P data integration. We developed the approach of Unspecified Ontologies (UNSO) that instead of providing mapping mechanisms between various ontologies, inherently supports dynamic organization of users (as providers of particular data objects) in a HyperCuP-like graph structure. The following section discusses the details of UNSO and the stages of generalizing the regular notion of ontologies to UNSO.

4 Generalization of Ontology to an Unspecified Ontology

In this work we propose a novel approach to resolve the issue of global ontology that should from one hand grow dynamically with no limited range, from other hand to be used and updated in a fully distributive way. Initially, let us characterize the structure of ontologies and their usage in semantic routing.

As a data structure, each ontology is considered as a vector, whose slots correspond to the attributes of the objects being described. As such, the range of values for each slot is the set of possible values of the respective attribute. For example, consider a simple ontology for cars domain, where the objects are described by a vector containing three slots only: [*manufacturer* | *engine_volume* | *year_of_production*]. Let us consider each slot having the following range of predefined values [*manufacturer={Ford, Mercedes, Jaguar}* | *engine_volume={1000-1500, 1500-2000, 2000-2500}* | *year_of_production={new, 2000+, 1990+}*]. Clearly, semantic P2P system based on this ontology will generate a hypercube containing at most *27* nodes, each having up to *6* logical neighbors.

A generalization of a fixed ontology to the Unspecified Ontology is performed at the following levels (referring to the above example of cars ontology):

- The set and the range of possible values is made unlimited by operating on them a fixed-length hashing instead of defining a set of predefined values. For example, in the above ontology we use hashing to a range of size three, mapping new values to their corresponding positions. For example, object [*manufacturer:BMW* | *engine_volume:3000* | *year_of_production:1987*] is mapped to [*hash(BMW)* | *hash(3000)* | *hash(1987)*]. Therefore, the same *3*-dimensional cube is now used to hold the objects, whose values were not anticipated by the slots of the predefined ontology. Moreover, the users can independently insert new objects in a fully distributed way, using the hashing above mechanism.
- More than one vector is used in the ontological description of an object, obtaining a *multi-layered* ontology that is *hierarchical* instead of using one *flat* vector. Hierarchical ontology produces a hypercube, whose nodes are recursively constructed of another hypercubes. This structure is further referred as a *multi-layered hypercube* (MLH). For example, consider a *3*-layered ontology with three vectors [$attr_{11}$ | $attr_{12}$ | $attr_{13}$] + [$attr_{21}$ | $attr_{22}$ | $attr_{23}$] + [$attr_{31}$ | $attr_{32}$ | $attr_{33}$] (where $attr_{ij}$ refers to slot number j in ontological layer number i) with two possible values for each slot. Such ontology generates a hypercube with *8* nodes, where each node recursively contains another hypercube (figure 3). This structure should be compared to

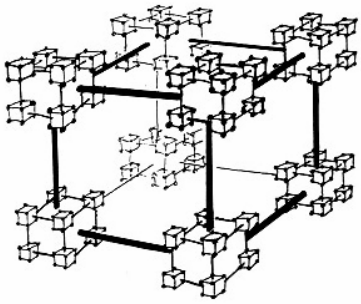

Fig. 3. Multi-Layered Hypercube (MLH)

a *512*-nodes hypercube, had we used one flat vector for the whole ontology. Obviously, hierarchical representation of the ontological vectors results in smaller and denser hypercubes, alleviating the connectivity maintenance of the hypercubes in pure decentralized P2P environment.

- The vector can dynamically grow by letting the users to add new levels to the ontology. An unspecified vector (ontology-based description) is formed by *attribute:value* pairs. Two different hashing functions are used to map the unspecified vector to the MLH. The first maps the *attributes* to the respective ontology slots, while the second maps the *values* to the numeric values of the slots. Thus, $hash_1(attribute)$ determines the dimension of the MLH, while $hash_2(value)$ determines the numeric value in this dimension. For example, consider the following description [*manufacturer:ford* | *engine_volume:1600*]. It is mapped to the underlying MLH by applying $hash_1(manufacturer)$ and $hash_1(engine\_volume)$ to obtain the slot numbers (MLH dimensions), and $hash_2(ford)$ and $hash_2(1600)$ to determine the numeric values in these dimensions.
- The users can specify a different number of unspecified slots. When an unspecified vector is mapped to an existing MLH, its slots are extended to the dimension of the MLH. For example, unspecified vector [*manufacturer:ford*] is extended to [*manufacturer:ford* | *engine_volume:1600*], when inserted to an MLH of dimension *2* from the above example. If the inserted object contains more slots than the number of dimensions in the current MLH, the vectors in the relevant hypercube are updated to accordingly contain more slots. We assume that upon long enough period of time and large enough number of object descriptions accumulated in the system, these updates will not be frequent, thus, they will be feasible. Note that exploiting two different hashing functions allows ignoring the order of the attributes in the unspecified descriptions.
- To distinguish between different objects having the same attributes (e.g., both cars and bicycles have attributes such as *color*, *wheels_type*, *number_of_gears* and so on), we require the ontological descriptions to contain so-called *universal,* or *specified slots*. A slot is referred as universal if it can be applied to as many objects as possible, and its values form a meaningful separation between the objects. For example, a universal attribute useful to separate between many objects is *size*, reflecting the size of the object comparing to a well-known standard. This allows distinguishing between the descriptions of cars and bicycles having the same attributes. It should be stated that we rely on a common sense and common patterns of thinking in specifying the values of the universal slots.
- The last $attribute_i:value_i$ pair in description must be a *discriminative attribute*, i.e., its should be unique for any given object. For example, consider an IP address of the user, or an ID number of the ad as such discriminative attributes. This allows two ads (either identical, or distinct, but accidentally mapped to the same node) to be separated according to the value of the discriminative attribute.

One of the heaviest steps in the management of the hypercube-like graph structure is the extension of the hypercube and the stored vectors as a result of a new slot mentioned in the ontological description of an object. Although these extensions are theoretically possible at any point of time, we conjecture that they mostly occur at the initial stages of graph construction. This is explained by the observation that large

number of new slots will be introduced during the initial insertions of objects, while at the latter stages the probability of introducing a new slot (unless introduced maliciously) that was not encountered yet, will decrease. As a result, the hypercube-like graph will stabilize at the initial stages of objects insertions and the extensions will be infrequent. This behavior is well-known in information search studies, where "*as a user obtains more information..., the probability of soliciting information from an additional source is likely to decrease*" [18]. Our conjecture regarding the similar behavior of UNSO slots is verified in the experimental part.

The above steps of extending a fixed specified ontology to the Unspecified Ontology are illustrated in figure 4. Predefined attributes and values of a fixed ontology are mapped to the underlying hypercube graph. On the contrary, in UNSO the number of <*attribute*$_i$:*value*$_i$> pairs in the unspecified description is unlimited. Thus, UNSO dynamically generates a hypercube-like graph structure, where each node is recursively constructed of another hypercube.

The generated MLH acts as a convenient infrastructure for the operations proposed in HyperCuP. When an object is inserted by a user to the MLH, it is forwarded towards its proper position. There, the user connects to its logical neighbors and starts functioning as an MLH node. Disconnection of a user causes another user to take the responsibility for the node of the disconnected user. Semantic routing in the MLH consists of a series of routings to the appropriate node in the current-layer hypercube

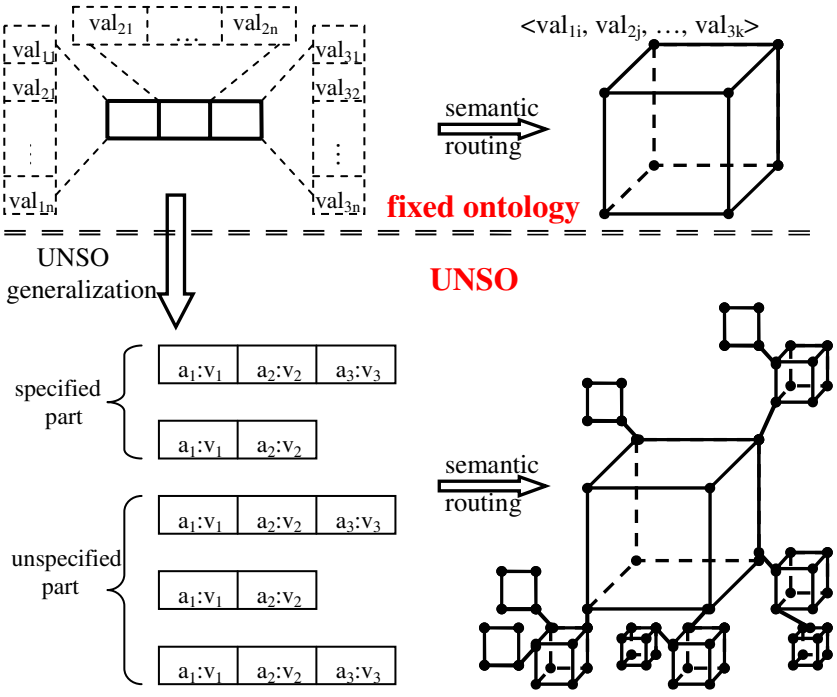

Fig. 4. Generalization of the Fixed Ontology to the Unspecified Ontology

and *deepenings* to the appropriate next-layer hypercube. Note that in the MLH the routing is performed according to Plaxton routing algorithm, i.e., every network hop corrects one digit of the address.

Important property of UNSO data structure is a property of *implicit locality* (basically, inherited from HyperCup). Both in HyperCup and in UNSO, the location of a user in the underlying MLH is determined according to the contents provided by the user. For example, consider two users providing similar (but not identical) data objects, described by the respective unspecified vectors: <$attribute_1$:$value_1$, $attribute_2$:$value_2$, $attribute_3$:$value_3$> and <$attribute_1$:$value_1$, $attribute_2$:$value_2$, $attribute_3$:$value_4$>. Clearly, mapping of both vectors to the underlying MLH will produce identical numeric coordinates within the dimensions corresponding to $attribute_1$ and $attribute_2$, whereas the coordinates within the dimension of $attribute_3$ will be different. As a result, the distance between the above pair of data objects will be lower than the distance between arbitrarily chosen pair of data objects. This can be explained by the observation that the distance between the locations of objects, whose descriptions overlap in a subset of <$attribute_i$:$value_i$> pairs, will be lower than the distance between the locations of objects with non-overlapping descriptions. We refer to this property of UNSO as implicit locality, since providers of similar contents are mapped to close locations in the underlying MLH graph.

Although the implicit locality property seems to contradict the load balancing property of hashing mechanism, this is not exactly the case. The insertion of the data objects into the MLH is performed through the mapping of <$attribute_i$:$value_i$> pairs, such that $attribute_i$ is mapped to one of the MLH dimensions, and $value_i$ to the numeric value (coordinate) within the dimension. Since UNSO exploits hashing-based mapping mechanisms, the distribution of values among any given MLH dimension will be uniform, and the load of MLH nodes will be similar. Although implicit locality property might lead to high-populated regions of the MLH, the number of real (not simulated) nodes in these regions will be also higher, and they will keep proper functionality under the conditions of higher load.

Another issue deals with the mapping of values from continuous domain to the MLH dimensions' coordinates. Although hashing-based mapping is suitable for mapping the values of discrete or non-enumerable attributes (e.g., *color*, *manufacturer*, or *model*), it is inappropriate for the values of continuous attributes (e.g., *year_of_production*, *price*, or *engine_volume*) since hashing-based mapping of continuous attributes loose their original inherent order relation. To resolve this issue, we replaced the hashing-based mapping of continuous attributes with a simple *bucketing* of the values. For example, consider a mapping of *price* attribute, whose values range between *0* and *100,000* to a dimension with *10* possible numeric values. In this case, prices between *0* and *10,000* will be mapped to numeric value *0* within the coordinate, prices between *10,000* and *20,000* – to *1*, and so forth. By doing so, we facilitate answering range and approximate queries over the continuous coordinates, since close values are mapped to the same (or neighbor) buckets, and the order relation of values between the buckets is also kept.

In summary, we would like to stress the main advantages of the Unspecified Ontologies mechanism and the underlying MLH graph of UNSO over the approach proposed by HyperCuP:

- Lack of centralized ontology – the mapping of an object to its position in the MLH is performed through hashing-based mechanism, instead of using a fixed predefined ontology.
- Contents flexibility – new types of objects and attributes can be easily added to the system, not requiring any updates of the ontology.
- Unlimited range of attributes and values – the unspecified part of the ontology allows the users to mention relatively free list of attributes and values in the ontological descriptions of the objects.
- Insignificant order of attributes – the unspecified part of the ontology does not require any order of the mentioned attributes, allowing high flexibility in further search of the data objects.
- Implicit locality – similar descriptions (i.e., similar objects), are mapped to adjacent positions within the MLH. Thus, similar objects are autonomously clustered without any explicit clustering mechanism.
- Approximated search – property of implicit locality in UNSO allows employing approximated searches, finding the objects similar (but not identical) to the object being searched.
- Partial searches – a search, containing a subset of the required object attributes can be conducted, finding a wider range of relatively similar objects.
- Multi-layered structure – increases the independence of the data management, as an update a particular hypercube layer does not affect the rest of the layers.

Compared to the existing DHT-based systems, UNSO proposes functionalities that were not supported earlier. Hashing mechanism of DHT systems is based on a single key. As a result, successful search in DHT-based systems requires exact matching of the keys. This is unreasonable when the key is a natural language description of an object. Conversely, UNSO handles keys represented by a list of attributes and their respective values. Thus, UNSO facilitates partial search of a subset of object attributes, whereas the order of the specified attributes is insignificant. Hence, data management and search mechanisms of UNSO are more flexible, than the parallel mechanisms of DHT-based systems.

Comparing UNSO to HyperCuP [32] yields another important advantage of UNSO. HyperCuP is based on a fixed predefined ontology that must be explicitly used by all the users. This is a severe limitation, as the users are forced to share and use it when describing their objects. On the contrary, UNSO allows the users to provide relatively free description of objects in form of $attribute_i$:$value_i$ pairs list. As well, new objects and attributes can be easily added to UNSO, not requiring update of the ontology and the update distribution to maintain consistency. Multi-layered structure of UNSO provides an efficient mean to resolve hashing collisions, since the attributes and their values are used to distinguish between objects from different domains.

5 UNSO Implementation Details

This section discusses the details of UNSO implementation. A multi-layered UNSO model, similar to the model, described in the previous section, was implemented. The

primary layer of UNSO contains the hypercube constructed by the specified part of the ontological descriptions of objects. The secondary MLHs are originated by the unspecified parts of the data objects' descriptions.

The specified part of UNSO is the subset of object attributes that should by explicitly mentioned by the users when describing the object. As the target application objects are general-purpose E-Commerce ads, the slots of the specified part should be applicable to as wide as possible range of objects. Clearly, these attributes are the *universal* attributes of the objects, such as size, weight, price, material and so forth. Note that here we refer to real-life tradable objects only, such as cars, apartments, books, tickets and others, which are the matter of E-Commerce ads.

The following set of the universal attributes was chosen to serve as the slots of the specified part of the ontology:

- *Product* – the name of the object, i.e., a noun describing the class of objects the described object belongs to. For example, car, book, telephone and so forth.
- *Relative size* – the size of the object. Since the size is a relative concept, we provide a standard size, such that the size of the described object is measured with respect to it. The standard is chosen to be an average size of a human being, and the objects are graded from 'very small' to 'very big' in comparison to it. For example, pen or mobile phone are 'very small', while house or truck are 'very big'.
- *Usage range* – the physical distance from the current location of the user, where the object is operated. As this concept is also a relative one, a standard for the possible values is provided. The usage distance is defined as 'very close', if the object is operated "at arm's reach" (i.e., in the room or indoor), and 'very far' if the distance of operation is roughly unlimited. Thus, housewares or furniture are 'very close' objects, while cars or optical equipment are 'very far'.
- *Price* – the price of the object, i.e., the sum the seller expects to get, or the sum the buyer is ready to pay for it. Price is also a universal attribute as it separates between different types of objects, such as houses and cars (although it may fail to separate books from CDs).

The *attribute:value* pairs mentioned by the user in the ontological description of an object, determine its position in the underlying MLH. The dimension of the primary hypercube equals to the number of slots in the specified part of the ontology. Thus, the above four slots generate a *4*-dimensional primary hypercube. The number of dimensions in the secondary MLHs is theoretically unlimited. Practically, a number of attributes mentioned by the users in the unspecified part of ontological descriptions is limited and quickly converges to its upper bound. Therefore, HyperCuP approach, where the connected users simulate the missing nodes, allows to maintain connectivity in the generated MLH.

In HyperCuP, the mapping of objects to the hypercube was performed according to the values of the ontology slots. On the contrary, in UNSO, the mapping is performed through a set of simple hashing functions operating with the numeric codes of the characters used in the values of ontology attributes. Although smarter hashing mechanisms could provide better distribution [28] and keep the locality and order relations between the values [17], in this implementation we chose a simple hash functions only to validate the applicability of the proposed approach.

Thus, the range of possible values (further referred as the *coordinates range*) k in the specified slots *product* and *price* is determined by the range of the hashing functions. Since for *relative size* and *usage range* slots the set of values is restricted to {*very small, small, medium, big, very big*}, the total number of nodes in the primary hypercube is practically limited by $25k^2$, where k is the coordinates range.

One of the advantages of hashing-based mapping mechanism is in a uniform distribution of ads among the MLH allowing to uniformly partition the load of the users. For example, for *k=17*, an ad [*product:car* | *relative_size:very big* | *usage_distance: very far* | *price:45000*] is mapped to [*0* | *4* | *4* | *13*] in the primary hypercube, while an ad [*product:television* | *relative_size:medium* | *usage_distance:very close* | *price:400*] is mapped to [*15* | *2* | *0* | *2*]. However, for *k=11* the same ads are mapped to other locations, respectively [*7* | *4* | *4* | *8*] and [*6* | *2* | *0* | *1*].

As for the unspecified part of the ontology, the format is a list of *attribute:value* pairs, where neither the *attributes*, nor the *values* are limited by any predefined ontology. For example, consider the following ontological (both specified and unspecified) description of a car: [*product:car* | *relative_size:very big* | *usage_distance: very_far* | *price:45000*] + [*manufacturer:BMW* | *color:red* | *mileage:5000*]. The unspecified part [*manufacturer:BMW* | *color:red* | *mileage:5000*] is mapped to a position *(5, 8, 4)* if the attribute 'manufacturer' is mapped to the coordinate number *1* and the value 'BMW' is mapped to the value *5*, 'color' is mapped to coordinate *2* and 'red' to value *8*, and respectively 'mileage' is mapped to coordinate *3* and '5000' to value *4*.

Note that two different ads are mapped to the same node either if their ontological vectors are identical, or the mappings of their *attributes* and *values* completely overlap. In this case the IP address of the user inserting the ad is chosen to act as the *discriminative* attribute distinguishing between the ads. Since the IP address is not a real attribute of the described object, it acts as a discriminative attribute only and is disregarded when displaying the search results.

Every new attribute, mentioned in the ads from a particular domain, increases the total dimension of the respective MLH. For example, consider the domain of car ads with a single ad [*product:car* | *relative_size:very big* | *usage_distance: very_far* | *price:45000*] + [*manufacturer:BMW* | *color:red* | *mileage:5000*] from previous example. Inserting a new ad, mentioning two unspecified attributes that were not mentioned yet (e.g., *gear* and *number_of_doors*), causes the secondary hypercube to expand to a *5*-dimensional hypercube. So, the original ad is remapped to *(5, 8, 4, *,*)*. However, analyzing relatively large corpus of car ads shows that the inserted ads contain in total *12* different attributes only, while most of the ads contained only *3-4* pairs of attributes and values. Thus, most of the hypercube expansions occur during the initial phases of the MLH construction. In this stage the MLH still contains relatively low number of ads and therefore the update is not expensive. Conversely, in a hypercube containing many ads, the updates are infrequent.

A possible drawback of the proposed idea is unclarity and ambiguity in supplying the values of the ontology slots. For instance, one user might define a car as *big*, while the other one might search it as *very big*. In this case the search is mapped to a different position in the hypercube, and the matching can not be identified. We conjecture that most of the users use similar *patterns of thinking* while describing the objects. If this is true, most of the ads describing the same objects are mapped to the same position in the primary hypercube, forming *clusters* of similar ads (figure 5).

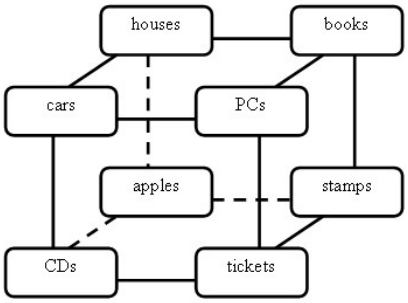

Fig. 5. Cluster of Ads in a 3-Dimensional Primary Hypercube

To resolve the issue of semantic heterogeneity, i.e., different terms with the same semantic meaning in the descriptions of objects, values of the ontological slots are standardized using WordNet [11]. In WordNet, English nouns, verbs, adjectives and adverbs are organized into *synonyms sets*, each representing a single semantic concept. For each concept, the synonyms' set is sorted according to the usage frequency. In our implementation, we employed a simple semantic standardization, replacing the original term, mentioned in object description, with its most frequent synonym. Thus, similar but not identical terms mentioned by the users, are replaced with a single representing term. For example, terms such as "automobile", "machine", or "motorcar", are replaced with their most frequent synonym "car".

Another practical issue leading to the semantic heterogeneity might be the use of hyponyms or hypernyms (respectively, terms that are more specific, or more generic than popular, standard term implied by most of the users) in the descriptions of objects. For example, a user might use '*sport car*' or '*coupe*' hyponyms or '*motor vehicle*' hypernym while describing the above mentioned red BMW *car*. Although current implementation of UNSO can not handle such standardization, it can be easily accomplished through hyponym and hypernym links of the WordNet. This will allow the standardization process to scan child and parent nodes in addition to currently scanned sibling nodes of WordNet graph of nouns, and therefore will improve the retrieval capabilities of UNSO.

To balance the communication load across the system, a modified variant of Plaxton routing algorithm [25] is implemented. Plaxton algorithm routes a message by correcting each time a single digit of the address. In the original algorithm the order of address digits corrections is known, i.e., the most significant digit of the address is corrected first, and the least significant digit is corrected last. Simultaneous routing of multiple messages to/from the same area (cluster) might lead to a communication bottleneck. To resolve this issue, we randomized the order of address digits corrections. This is further referred as a *randomized* Plaxton routing.

Over the hypercube-like graph topology (which is a subclass of expander graphs [24]), Plaxton routing algorithm corrects one digit of the address at each network hop. Since the bits of HyperCuP neighbor nodes differ in exactly one digit of the address, and each correction modifies only a single digit of the address, the order of the digit corrections can be shuffled. For example, consider correcting an address of length *logN*, where *N* is the maximal number of users in the hypercube. The number of

possible correction orders that can be applied, i.e., the number of possible routes, is *(logN)!*. This number allows high routing flexibility in the hypercube-like expander graph topology and improves the communication load balancing.

On the first view, Plaxton routing algorithm seems to be very inefficient form of routing, since correcting a multi-dimensional address in a MLH of total dimension *d* with coordinates range *k* will require *O(kd)* network hops. However, this observation will not hold in practical P2P systems which will be highly sparse, i.e., the number of real connected nodes will be significantly lower than the number of possible locations in the underlying MLH. Due to the sparsity of the MLH, a number of nodes will be simulated by a single user and as a result, the number of network hops required for the query routing will be significantly decreased. In any case, the above randomized Plaxton routing allows to better distribute the communication load among the connected users.

The system prototype implementation includes Graphical User Interface (GUI) enabling the users to launch search queries (figure 6). The GUI contains fields for inserting the values of the specified slots, textual fields for the unspecified attributes and values, and textual area for displaying query results.

In a typical search scenario, the user starts from filling-in the values of the specified slots of the ontology. Assuming common patterns of thinking, the system finds a node in the primary hypercube, which stores the ads from the relevant domain. For a large enough corpus of ads, the number of the matching ads might be too high. Thus,

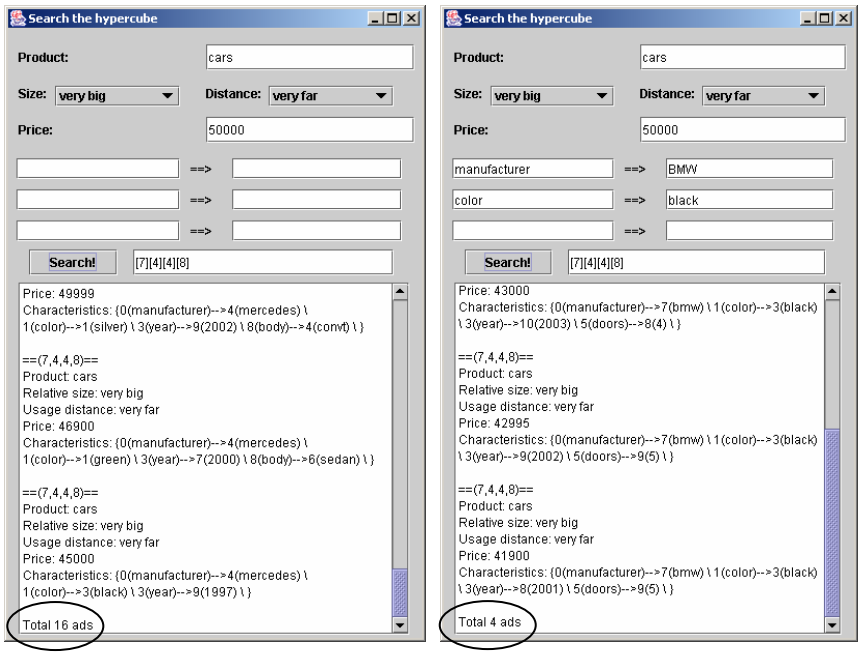

Fig. 6. Typical Search Scenario (a) Only the slots of the fixed part of the ontology are specified; (b) In addition, two unspecified slots are specified

the user needs to refine the query by mentioning the unspecified attributes and their values. As a result, the system filters out the irrelevant ads and displays to the user a shorter list of ads.

Consider the search scenario, illustrated in figure 6. In the first query (figure 6a), the specified slots [*product:car* | *relative_size:very_big* | *usage_distance:very_far* | *price:50000*] only are filled-in. The query is mapped to a node [*7* | *4* | *4* | *8*] of the primary hypercube (for *k=11*). As no unspecified attributes are mentioned, the number of returned ads is *16* (out of *34* matching ads in the corpus), which is relatively high. In the second query, the user expands the description using two unspecified attributes: *manufacturer* and *color*. The search for [*product:car* | *relative_size: very_big* | *usage_distance:very_far* | *price:50000*] + [*manufacturer:BMW* | *color:black*] is mapped to [*7* | *4* | *4* | *8*] + [*7* | *3* | * | * | *]. Note that since the secondary MLH consists of a single *5*-dimensional hypercube, the system automatically expands the address through inserting wildcards instead of the attributes that are not mentioned in the search. The use of unspecified attributes filters the irrelevant ads, returning a smaller set of *4* matching ads only (figure 6b).

6 Experimental Results

To conduct the experiments, a corpus of both supply and demand E-Commerce ads was downloaded from *http://www.recycler.com*. The corpus of supply ads contains *1272* ads from *14* different categories. Before inserting the ads into the system, they were manually converted to the form of ontological descriptions. For example, the following ad “Philips 50FD995 50" plasma television, brand new, $4800” was converted to the following unspecified ontological description [*product:television* | *relative_size:medium* | *usage_distance: very_close* | *price:4800*] + [*manufacturer:Philips* | *model: 50FD995* | *size:50"* | *material:plasma* | *condition:brand new*]. The conversions were done as close as possible to the original contents of the ads, to mimic the insertions by naïve users. Corpus of *136* demand ads was automatically built by modifying a subset of the attributes and values mentioned in the supply ads.

The following experiments were conducted. Initially, we evaluate the Information Retrieval metrics of precision and recall [37] through launching queries for the objects from the demand ads. Then we validate the property of locality, namely the distance between similar and dissimilar ads in UNSO. Finally, we measure the scalability and stability of UNSO through evaluating the performance of the system for gradually increasing number of ads in the system. The results of the experiments are presented in the following sub-sections.

6.1 Information Retrieval Metrics

This experiment was designed to evaluate two traditional Information Retrieval accuracy metrics: *precision* and *recall* [37]. In context of publish-locate application, precision is computed as the number of relevant ads in a node divided by the total number of ads there. Similarly, recall is computed as the number of relevant ads in a node divided by the total number of relevant ads in the system. For example, consider *100* cars ads inserted to the system. User looking for a car launches a query, and receives

80 ads. *60* out of them are cars ads, while the rest are ads from other application domains. In this case the precision is *60/80=0.75*, and the recall is *60/100=0.6*.

An ad might be included in a query results due to two reasons: it might either be an ad that really satisfies the constraints posed by the query, i.e., mentions the terms that were searched in the query, or it might be irrelevant ad that was accidentally mapped to the same node. To accurately measure the values of precision and recall, relevance metrics should be explicitly defined. In this work, the relevance of the returned ads is automatically determined through comparing the attributes and the values mentioned in the query with attributes and values mentioned in the ads returned by the system.

The values of precision and recall are measured for different coordinate ranges k (hashing function range) of the UNSO slots, varying from *1* to *100*. For each value of k we compute the average recall and precision among launching the *136* demand queries. Recall and precision are computed through computing the number of relevant ads and the total number of ads in the nodes the queries are mapped to, and the total number of ads relevant from the given domain.

The chart in figure 7 shows the average values of precision and recall as a function of the coordinates range k. The dashed curves show the precision values, while the continuous curves stand for the recall. Both the precision and the recall are measured twice: for the original terms specified in the ads (the brighter curves), and after standardizing the values of the ontology slots with WordNet (the darker curves).

In general, both the probability of hashing collisions and the number of irrelevant ads, accidentally mapped to a node, decrease with the increase of k. Thus, the precision increases with the coordinates range k. It can be clearly seen from the chart that the standardized results outperform the original results. This holds for any coordinates range k, and the standardized precision values asymptotically converge to *1* (maximal precision,

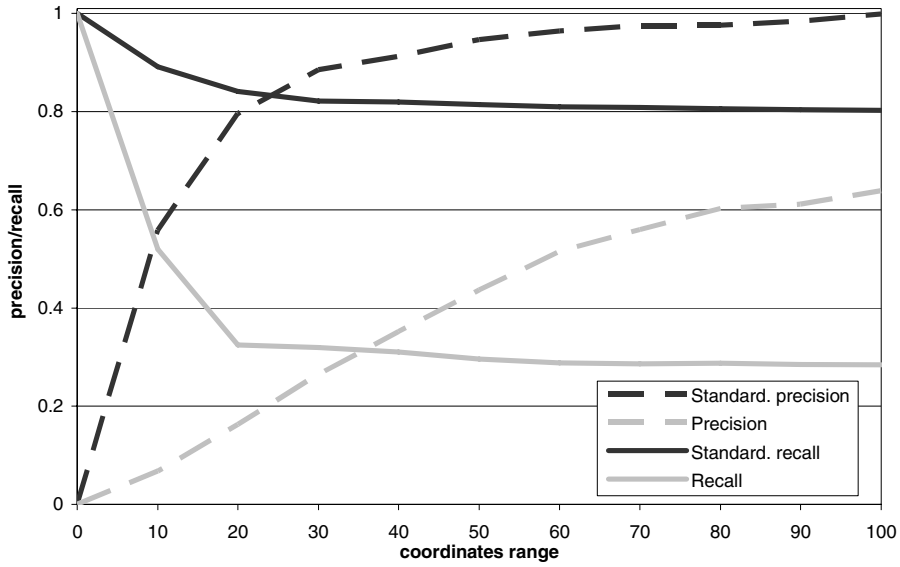

Fig. 7. Recall and Precision as a Function of the Coordinates Range k

if all the ads returned by a search are relevant) starting from relatively low values of k ($k>30$). The observation that the standardized precision values are higher is reasonable, since WordNet replaces similar terms mentioned in the ads with a single representing term. Thus, the number of different attributes and in particular the number of ads accidentally mapped to a node monotonically decreases with k, whereas the precision monotonically increases.

The original recall values of the system are relatively low, approximately *0.29*. This is explained by the observation that without the WordNet standardization the users mention different terms to describe the same semantic concepts. Consequently, the searches succeed in finding only the ads that mentioned the appropriate term (or the ads that were accidentally mapped to the same position). Using WordNet, the recall values are significantly higher, roughly *0.8*. Note that initially the recall values decrease when the coordinates range k increases. This happens due to the fact that the probability of similar ads mentioning different terms to be accidentally mapped to the same node is relatively high for low values of the coordinates range k. For example, for $k=1$ all the ads are mapped to the same location, as the MLH basically consists of a single node.

Obviously, there is a clear trade-off between the coordinates range k and the accuracy (in terms of recall and precision) of the system. From one hand the chart shows that a linear combination of recall and precision improves for higher values of k, as the recall converges starting from relatively low values of k, and the precision increases. However, practical maintenance of large enough MLHs is complicated for a low number of connected users, due to the sparsity causing the connected users to simulate high number of 'missing' nodes.

For example, let us compare two possible coordinate ranges: $k=30$ and $k=80$. The values of recall and precision for these values of k are close: the precision is respectively *0.89* and *0.97*, while the recall roughly remains unchanged, *0.81*. But, the estimated size of the primary hypercube only increases from *22,500* to *160,000* nodes. Thus, every connected user is forced to simulate about *7* times more nodes, resulting in a significantly higher communication loads and maintenance overheads of the connected users. Therefore, we conjecture that the optimal value of k for a moderate number of connected users and inserted ads is between *20* and *30*. For such values of k, both precision and recall values are above *0.8*, while the size of the primary hypercube remains reasonably small.

6.2 Locality

An important metric of quality of a system using semantic routing is locality, i.e., mapping of similar ads to close positions in the underlying MLH. To verify this property, a subset of *128* supply ads was randomly chosen. For each chosen ad X, we denote by X'_j a modified ad, where the values of j attributes are changed. Both X and X'_j are inserted into the system and the distance d between their positions is computed.

The distance d between the positions of two ads X and X'_j is defined as a sum of their distances (i.e., differences in the numeric coordinates) in each one of the dimensions. For example, the distance between the positions (*1, 3, 5, 7*) and (*8, 6, 4, 2*) in a *4*-dimensional hypercube is $|(1-8)|+|(3-6)|+|(5-4)|+|(7-2)|=16$. In the MLH topology, distance between two positions is the sum of the distances in each one of the layers.

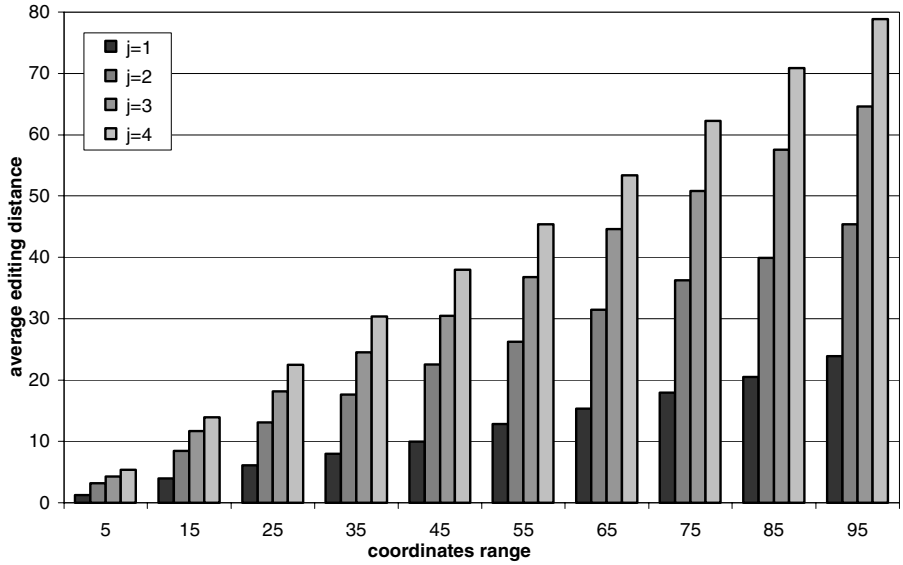

Fig. 8. Average Distance as a Function of a Number of Changes (*j*) for Different Values of the Coordinates Range *k*

The chart in figure 8 shows the average distance *d* over the chosen *128* ads as a function of *j*, computed for the coordinates range *k* varying from *1* to *100*.

Each quadruplet of bars in the chart shows the average distance between the ads for *j=1*, *2*, *3* and *4* (the leftmost column stands for one change, and the rightmost for four changes). It can be clearly seen that for any value of the coordinates range *k,* the distance increases with the number of changes inserted to the ads. For example, for *k=55*, the distance for one change is *12.8*, while for four changes it is *45.4*. The results show that the property of locality holds in UNSO and the average distance between two similar ads (small number of changed values) is lower than the average distance between two different ads (higher number of changed values).

The chart also shows that the distance *d* for a given *j* increases with the increase of the coordinates range *k*. The higher is the value of *k*, the wider is the range of possible coordinates the ad can be mapped to, and the higher is the distance. However, the growth of the average distance is linear with *k*. This observation holds for all the values of *j*, i.e., it is true for different magnitudes of the distance.

In the discussion on precision, we concluded that the best precision is achieved for high values of *k*. The linear growth rate of the distance shows that precision improvement will not negatively affect the distance and the property of locality will hold for different values of coordinates range *k*.

6.3 Scalability

Scalability is one of the most important properties of P2P systems, and certainly one of their biggest advantages of P2P over the Client-Server approach. We define it as an

indicator of a system's ability to maintain quality performance under an increased load. In UNSO, "maintenance of quality performance" means reasonable MLH data management and efficient query routing under conditions of heavy load. The meaning of "heavy load" can be interpreted in a few ways. It can be referred as a high number of ads in the system, communication overload, high rate of users' joins and disconnections, and so on. This experiment was designed to evaluate the scalability of UNSO under conditions of high communication overload in a particular area of the MLH.

To create a high number of routing operations in a particular area, we insert into the system two sets of ads (about *120* ads in each one): housewares ads and telephones ads. These sets generate two clusters, further denoted as *P* and *Q*. We gradually increase the number of ads in each cluster and simulate the communication overload by simultaneously routing queries from *P* to *Q* and from *Q* to *P*. For each ad in *P* (and in *Q*) we randomly select an ad in *Q* (and respectively in *P*), and simulate a routing operation between the positions of the ads. In the experiments we measure the maximal number of queries routed through a single node. The measures are conducted for a number of coordinates ranges *k*, such as *k=13*, *17*, *23*, *29*, and *37*. As the randomized Plaxton routing is not deterministic, the experiments are repeated *1000* times. The chart in figure 9 shows the average results.

The chart shows that starting from approximately *36* ads (*30%* of the maximal size of the sets) in *P* and *Q*, increasing the number of ads does not affect the maximal traffic in the system, i.e., the number of simultaneous routings increases without any significant change in the traffic load. This observation is explained by a relatively high connectivity of the underlying MLH structure (as a subclass of expander graph) and by the randomized variant of Plaxton routing algorithm that is exploited in

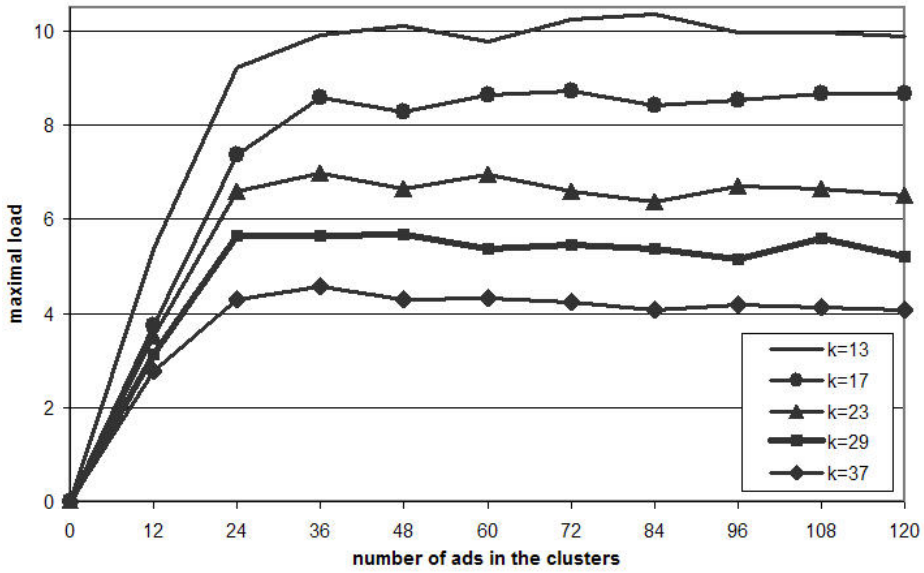

Fig. 9. Maximal Load in the Network Under Conditions of High Number of Simultaneous Routing Operations

UNSO. Note that this observation holds for different values of the coordinates range *k*. Moreover, the maximal load in the system decreases with the increase of *k*, as for high values of *k* the number of possible routing paths is higher and the communication overload of each node is lower.

This allows us to conclude that UNSO is scalable with respect to its functionality under conditions of communication overload resulting in a high number of concurrent routings. Thus, the proposed UNSO approach implemented over the MLH topology can work-out and function efficiently as a large-scale general-purpose E-Commerce application (e.g., launched over the Web).

6.4 Stability

In this experiment was designed to measure the "average size" of the generated structure as a function of the number of inserted ads. One of the key metrics for evaluating the size of the network is the characteristic path length. For two given nodes *A* and *B* in the MLH, a *path length* between their positions in the hypercube is defined as a minimal number of Plaxton corrections needed in order to reach *B* starting from *A*. The *characteristic path length* is defined as the average path lengths over all pairs of nodes in the system. In this experiment we gradually increase the number of ads inserted into the system and compute the characteristic path length. Since the ads inserted into the system are chosen randomly from the corpus of *1272* supply ads, for each number of ads in the system the experiment is repeated *1000* times. The average values of the characteristic path length are shown in figure 10.

The chart shows that the initial insertions of ads increase the characteristic path length by increasing the number of dimensions in the MLHs. However, starting from

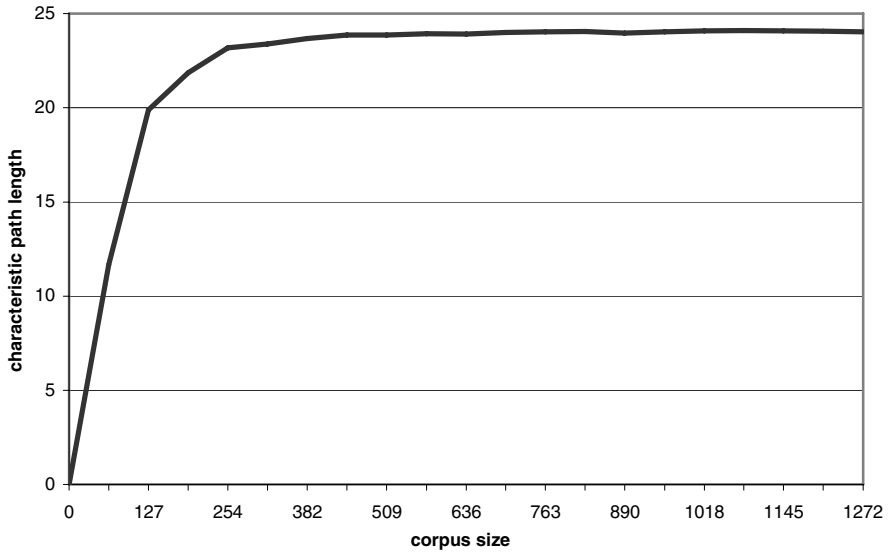

Fig. 10. Characteristic Path Length as a Function of the Corpus Size

approximately *20%* of the maximal corpus size, inserting ads has roughly no effect on the characteristic path length. This validates our assumption that the total cardinality of the set of attributes used for describing an object is final and bounded. Moreover, most of the attributes are used in the first descriptions that are inserted, while the rest of the descriptions contribute very few new attributes. Therefore, the proposed algorithm of the MLH expansion during the insertion of higher-dimensional ads does not lead to a significant overhead due to the expansion.

7 Conclusions and Future Work

In this work we proposed and evaluated a novel notion of Unspecified Ontology (UNSO). This concept refers to a non-fixed variant of ontological descriptions of the data objects. In comparison to the fixed predefined ontologies, UNSO facilitates more flexible way to describe the objects. Multi-layered hypercube graph (MLH) topology, supporting efficient semantic routing, is constructed based on UNSO. We designate the MLH to be used as an infrastructure for general-purpose E-Commerce applications over Peer-to-Peer network. This work focuses on the basic functionality of E-Commerce systems, i.e., publishing and locating the objects. It allows the users to publish their objects using ontological description, while other users are able to easily locate them.

Note, that using UNSO does not force the users to share or to use any single explicit ontology. The users can provide relatively free ontological descriptions of their data object by specifying the object attributes and their respective values. Instead of using a predefined ontological mapping, hashing-based mechanism is used to map the ads to their positions in the underlying MLH. Thus, the order of the attributes, mentioned by the user is insignificant, and if the attributes are properly recognized, they are mapped to the same hypercube coordinates.

Experimental results show that the performance of UNSO with respect to the Information Retrieval metrics is good. The precision of the system is high, and it increases with the increase of the MLH coordinates range. Although the recall values are moderate, they become very high as a result of performing a simple standardization with WordNet. Note that UNSO, unlike traditional Information Retrieval systems and search engines, demonstrates high values of both recall and precision at the same time. Smart semantic standardization tools and dynamic hashing mechanisms, uniformly distributing the ads among the MLH nodes, will assist in achieving a better performance with respect to the Information Retrieval metrics.

Performance of UNSO with respect to the P2P metrics is also good. As can be seen form the results of scalability experiments, the proposed system is highly scalable. For a relatively low number of concurrent routings, the system reaches its maximal communication load, and further routings do not increase it. This leads to a conclusion, that when implemented and launched as a large-scale data sharing P2P application, e.g., over the Web, UNSO-based systems will keep proper functioning under conditions of heavy load. Thus, we believe UNSO can serve as efficient infrastructure for real-life Web applications and it has the potential of opening Web E-Commerce activities to a larger community of P2P users.

The following directions of further research, based on the developed UNSO concepts, are of particular interest:

- Developing a statistical model for hierarchical management of the MLHs. It will simplify the maintenance of the MLH, shorten the characteristic path length, ensure semantic homogeneity in every level and inherently introduce smart error-correction mechanism. In addition, it will protect the hypercube-like graph against organized attacks, maliciously introducing new ontology slots.
- Improving the proposed simple standardization mechanism through integrating more sophisticated Natural Language Processing (NLP) tools recognizing the terms context during the semantic standardization. This will improve the Information Retrieval performance of UNSO.
- Discovering the importance (weights) of attributes and the distances between the different values of the same attribute. This will allow more accurate assessment of the distance between two nodes in the hypercube-like graph and will facilitate more accurate ranking of the results displayed to the user.
- Developing a smarter routing algorithm, dynamically generating the optimal routing path between two MLH nodes as a function of the changing workloads of the connected users.
- Adding other E-Commerce functionalities to the system. For example, real-life E-Commerce systems usually support sophisticated functionalities (e.g., public auctions and market clearings) rather then simple publish-subscribe functionality. We intend to investigate exploiting UNSO for supporting such functionalities.
- Implementing distributed P2P UNSO client, launching it over the Web and creating hypercube-like graph of real-life users, and performing large-scale experiments with large number of E-Commerce ads.

In summary, we believe that the unspecified and flexible nature of UNSO, jointly with its good data management capabilities, will facilitate UNSO usage not only for E-Commerce applications, but for any kind of publish-locate services.

References

[1] K.Aberer, M.Punceva, M.Hauswirth, R.Schmidt, "*Improving Data Access in P2P Systems*", IEEE Internet Computing, January-February 2002.

[2] K.Aberer, P.Cudre-Mauroux, M. Hauswirth, "*The Chatty Web: Emergent Semantics Through Gossiping*", In proceedings of the International World Wide Web Conference, Budapest, Hungary, 2003.

[3] E.Adar, B.Huberman, "*Free Riding on Gnutella*", Technical Report, Xerox PARC, 2000.

[4] S.Androutsellis-Theotokis, D.Spinellis, "*A Survey of Peer-to-Peer Content Distribution Technologies*", ACM Computing Survey, 36(4), pp. 335-371, 2004.

[5] S.Ayyasamy, C.Patel, Y.Lee, "*Semantic Web services and DHT-based Peer-to-Peer Networks: A New Symbiotic Relationship*", In proceeding of the Workshop on Semantics in Peer-to-Peer and Grid Computing, Budapest, Hungary, 2003.

[6] P.Backx, T.Wauters, B.Dhoedt, P.Demeester, "*A Comparison of Peer-to-Peer applications*", In proceedings of Eurescom Summit, Heidelberg, Germany, 2002.

[7] P.Bernstein, F.Giunchiglia, A.Kementsietsidis, J.Mylopoulos, L.Serafini, I.Zaihrayeu, "*Data Management for Peer-to-Peer Computing: A Vision*", In proceedings of the Workshop on the Web and Databases, Madison, Wisconsin, 2002.

[8] P.A.Bernstein, S.Melnik, "*Meta Data Management*", in Proceedings of the International Conference on Data Engineering, Boston, MA, 2004.

[9] A.Blum, T.Sandholm, M.Ziukevich, "*Online Algorithms for Market Clearing*", In proceedings of the ACM-SIAM symposium on Discrete algorithms, San Francisco, CA, 2002.

[10] I.Clarke, O.Sandberg, B.Wiley, T.Hong, "*Freenet: a Distributed Anonymous Information Storage and Retrieval System*", In proceeding of the ICSI Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, 2000.

[11] C.Fellbaum, "*WordNet - An Electronic Lexical Database*", MIT Press, 1998.

[12] N.Fridman Noy, M.A.Musen, "*PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment*", In proceedings of the National Conference on Artificial Intelligence (AAAI'2000), Austin, TX, 2000.

[13] T.R.Gruber, "*A Translation Approach to Portable Ontology Specifications*", Knowledge Acquisition Journal, 6(2), pp. 199–221, 1993.

[14] A.Gupta, D.Agrawal, A.Abbadi, "*Approximate Range Selection Queries in Peer-to-Peer Systems*", In proceedings of the Conference on Innovative Data Systems Research, Asilomar, CA, 2003.

[15] M.Harren, J.M.Hellerstein, R.Huebsch, B.T.Loo, S.Shenker, I.Stoica, "*Complex Queries in DHT-based Peer-to-Peer Networks*", In proceedings of the International Workshop on Peer-to-Peer Systems, Cambridge, MA, 2002.

[16] E.Hovy, "*Combining and Standardizing Large-Scale, Practical Ontologies for Machine Translation and Other Uses*", In proceedings of the International Conference on Language Resources and Evaluation, Granada, Spain, 1998.

[17] P.Indyk, R.Motwani, P.Raghavan, S.Vempala, "*Locality-Preserving Hashing in Multidimensional Spaces*", In proceedings of the Symposium on Theory of Computing, El Paso, TX, 1997.

[18] E.J.Johnson, W.Moe, P.S.Fader, S.Bellman, J.Lohse ,"*On the Depth and Dynamics of Online Search Behavior*", Management Science, 50 (3), pp. 299-308, 2004.

[19] J.Madhavan, A.Y.Halevy, "*Composing Mappings Among Data Sources*", In proceedings of the Conference on Very Large Databases, Berlin, Germany, 2003.

[20] P.Maes, R.H.Guttman, A.G.Moukas, "*Agents that Buy and Sell: Transforming Commerce as we Know it*", Communications of the ACM, 42(3), pp. 81-91, March 1999.

[21] D.S.Milojicic, V.Kalogeraki, R.Lukose, K.Nagaraja, J.Pruyne, B.Richard, S.Rollins, Z.Xu, "*Peer-to-Peer Computing*", Technical Report HPL-2002-57, HP Labs, 2002.

[22] W.Nejdl, B.Wolf, "*Edutella: A P2P Networking Infrastructure Based on RDF*", In proceedings of the World Wide Web Conference, Honolulu, HI, 2002.

[23] W.S.Ng, B.C.Ooi, K.L.Tan, A.Zhou, "*PeerDB: A P2P-Based System for Distributed Data Sharing*", In proceedings of the International Conferene on Data Engineering, Bangalore, India, 2003.

[24] D.Peleg, E.Upfal, "*Constructing Disjoint Paths on Expander Graphs*", in International Journal on Combinatorics and the Theory of Computing, 9, pp.289-313, 1989.

[25] C.Plaxton, R.Rajaraman, A.Richa, "*Accessing Nearby Copies of Replicated Objects in a Distributed Environment*", In proceedings of ACM SPAA, Newport, RI, 1997.

[26] S.Ratnasamy, P.Francis, M.Handley, R.Karp, S.Shenker, "*A Scalable Content-Addressable Network*", In proceedings of ACM SIGCOMM, San Diego, CA, 2001.

[27] S.Ratnasamy, S.Shenker, I.Stoica, "*Routing Algorithms for DHTs: Some Open Questions*", In proceedings for the International Workshop on Peer-to-Peer Systems, Cambridge, MA, 2002.

[28] R.L.Rivest, "*The MD5 Message-Digest Algorithm*", Request for Comments 1321, IETF Network Working Group, 1992.

[29] A.Rowstron, P.Druschel, "*Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems*", In proceeding of the International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, 2001.

[30] O.D.Sahin, A.Gupta, D.Agrawal, A.Abbadi, "*A Peer-to-Peer Framework for Caching Range Queries*", In proceedings of the International Conference on Data Engineering, Boston, MA, 2004.

[31] S.Saroiu, K.P.Gummadi, S.D.Gribble, "*Measuring and Analyzing the Characteristics of Napster and Gnutella Hosts*", Multimedia Systems, 9, pp. 170-184, 2003.

[32] M.Schlosser, M.Sintek, S.Decker, W.Nejdl, "*A Scalable and Ontology-Based P2P Infrastructure for Semantic Web Services*", In proceeding of the IEEE International Conference on Peer-to-Peer Computing, Linkoping, Sweden, 2002.

[33] I.Stoica, R.Morris, D.Karger, M.F.Kaashoek, H.Balakrishan, "*Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications*", In proceedings of ACM SIGCOMM, San Diego, CA, 2001.

[34] I.Tatarinov, Z.G,Ives, J.Madhavan, A.Halevy, D.Suciu, N.N.Dalvi, X.Dong, Y.Kadiyaska, G.Miklau, P.Mork, "*The Piazza Peer-to-Peer Data Management Project*", ACM SIGMOD Record, 32(3), pp.47-52, 2003.

[35] M.Uschold, "*Creating, Integrating and Maintaining Local and Global Ontologies*", In proceedings of the European Conference on Artificial Intelligence, Berlin, Germany, 2000.

[36] G.Wiederhold, "*Mediators in the Architecture of Future Information Systems*", in "Readings in Agents", pp. 185-196, Morgan Kaufmann Pubs., 1997.

[37] I.H.Witten, A.Moffat, T.C.Bell, "*Managing Gigabytes: Compressing and Indexing Documents and Images*", Morgan Kaufmann Publishers, 1999.

[38] B.Yang, H.Garcia-Molina, "*Designing a Super-Peer Network*", In proceedings of the International Conference on Data Engineering, Los Alamitos, CA, 2003.