The Role of Caching and Context-Awareness in P2P Service Discovery *

Christos Doulkeridis
Department of Informatics
Athens University of Economics and Business (AUEB)
Athens 104 34, Greece
cdoulk@aueb.gr

Vassilis Zafeiris
Department of Informatics
Athens University of Economics and Business (AUEB)
Athens 104 34, Greece
bzafiris@aueb.gr

Michalis Vazirgiannis
Department of Informatics
Athens University of Economics and Business (AUEB)
Athens 104 34, Greece
mvazirg@aueb.gr

ABSTRACT

Mobile terminals (cellular phones, PDAs, palmtops etc.) emerge as a new class of small-scale, ad-hoc service providers that share data and functionality via mobile web services' calls. In mobile service discovery, it is often the case that implicit contextual information such as the location of the requestor, time the query was submitted, capabilities of the involved devices have a significant impact on query evaluation and the quality of the results. On the other hand, lack of scalability and the well-known *single point of failure* problem call for distribution of service directories into several peers. In this paper, we present an approach for context-aware service discovery where service directories reside in a P2P architecture. We explore the role and benefits of context-awareness and caching query results in the service discovery process. We conduct extensive experiments coming up with guidelines for service directory design in a P2P context.

1. INTRODUCTION

Service discovery from handheld devices differs from traditional service discovery. The dynamic nature of terminals - such as heterogeneity, availability, mobility - that act as service providers and share their data, resources and functionality, make existing discovery mechanisms designed for stationary services problematic. Mobile devices, despite their fast technological advances, still impose restrictions with respect to processing power, storage space, energy consumption and bandwidth availability [4].

On the other hand, it is often the case that implicit information related to both the requestor and the provider device can have a significant impact on query processing and on quality of the results. For instance, consider the query "images of local monuments" submitted by a user currently walking near Acropolis in Athens using a device that can handle images of size at most 640x480 pixel and of at most 8-bit color depth. Apparently the user searches for low quality images of the monuments on Acropolis hill (such as Parthenon, Erehteion etc.). In this case, it is obsolete to consider images a) of other monuments that are far away from the query location and b) of quality higher that the capabilities of the requestor device. Thus, both the location of the requestor and the device capabilities have a significant impact on the query processing plan. Generalizing the above, such implicit - contextual - information related to the query and the device need to be taken into account to increase effectiveness of query evaluation and quality of the results.

The fundamental mechanism for web service discovery is a service directory, i.e. a registry of service descriptions that facilitate service discovery based on specific parameters. In mobile and rapidly changing environments, there is a need for enhanced directories that inherently provide advanced functionality. A context-aware service directory [1, 2] increases the precision and efficiency of service discovery by matching the context of the user submitting a request against the service's contextual information. Moreover, lack of scalability and the well-known *single point of failure* problem require distribution of service directories.

In this paper, we present an approach for context-aware service discovery where service directories reside in a P2P architecture. We explore the role and benefits of a) context-awareness and b) caching query results, in the service discovery process. The main focus of this work is to investigate how caching improves context-aware service discovery in a P2P setting. We conduct extensive experiments evaluating the performance of service discovery based on parameters such as context (i.e. location), cache size, query load distributions, starting threshold for caching, etc.

Our contributions lie in:

- proving that context-awareness and caching increase the efficiency of service discovery in a P2P architecture
- offering initial guidelines, resulting from our experiments, with regards to the design of the service discovery process

The rest of this paper is structured as follows: in Section 2 we present our approach regarding context in mobile service discovery. In Section 3, we discuss query evaluation, particularly caching, based on contextual queries for services in a P2P architecture of service directories. Section 4 presents the experimental results. Related work is covered in Section 5. In Section 6, we conclude the paper and describe future research directions.

2. CONTEXT-AWARE SERVICES

Context in service discovery for mobile computing is defined as: *"the implicit information related both to the requesting user and*

*Work partially supported by EU IST project DBGlobe (IST-2001-32645) under the FET Proactive Initiative on Global Computing for the co-operation of autonomous and mobile entities in dynamic environments

MDM 2005 '05 Ayia Napa, Cyprus

service provider that can affect the usefulness of the returned results" [2]. *Service context* can be the location of the service, its version, the provider's identity, the type of the returned results and possibly its cost of use. On the other hand, each user is characterized by a *user context* that defines her current situation. This includes several parameters like location, time, temporal constraints, as well as device capabilities and user preferences. During service discovery user context is matched against service context in order to retrieve relevant services with respect to context-awareness.

In our application scenarios, we assume that user queries are characterized by locality of interest, in the sense that users are usually more interested in "fresh" and nearby resources. Freshness is related to the temporal dimension and implies that newly created shared resources (images, files, etc) are more popular than outdated ones. The user's location is matched with the location of services, in order to retrieve nearest services. Moreover, contextual queries have special properties that seem to favor caching, in the sense that they present high frequency of occurrence in certain locations and time. For instance, right after an athletic event like the 100m final, many users who did not capture a nice photo may issue requests, while walking away, for such photos from other users in the surrounding area. In this way, caching of contextual requests' results is motivated.

2.1 Data Model and Architecture

A set of service categories, represented as graphs (see Fig. 1), form the service directory [1]. Context is represented as a set of key-value pairs. Keys are called *dimensions* and they can take one or more discrete values or range over a specific domain [9]. Each service category may contain different contextual dimensions, based on the type of services it includes. For example, image type or resolution is important for image sharing services, whereas for information services the charged cost probably plays the most important role. In this paper, we will focus on one service category (image retrieval services), but we can generalize and apply the results to all other categories within the service directory.

Services are represented as leaves of the graph and they are depicted by circular nodes (see Fig. 1). The path of the graph that starts from the root and leads to the service defines the *service context* (also called service description in this paper). Moreover, the morphology of the graph is similar to a tree, with the only difference that there may be leaves which belong to different subtrees and can thus be reached through more than one path. Such leaves correspond to services that are valid under different contexts. For instance, in Fig. 1, the service with identifier #1 returns images of type jpg and is published by Chris in different locations depending on the hour of the day (near Acropolis at 9 and 10 a.m. and near Plaka at 12:00 a.m.). Thus, the service context at 9 a.m. is formalized as [9]:

[location=Acropolis,time=09:00,image=jpg,user=Chris]

In this work we capitalize on MobiShare, an architecture, proposed by our group [10], that *"provides an infrastructure for ubiquitous mobile access and mechanisms for publishing, discovering and accessing heterogeneous mobile resources in a large area, taking into account the context of both sources and requestors"*. Numerous wireless network access points partition the geographical two-dimensional space into areas of coverage (cells), in a manner similar to traditional GSM networks. Mobile users carrying portable devices stroll around and share their data through services. A Cell Administration Server (CAS) is responsible for device monitoring and management within a cell. Each CAS maintains (among other modules) a context-aware service directory (CASD) that stores information about all published services in the corresponding cell.

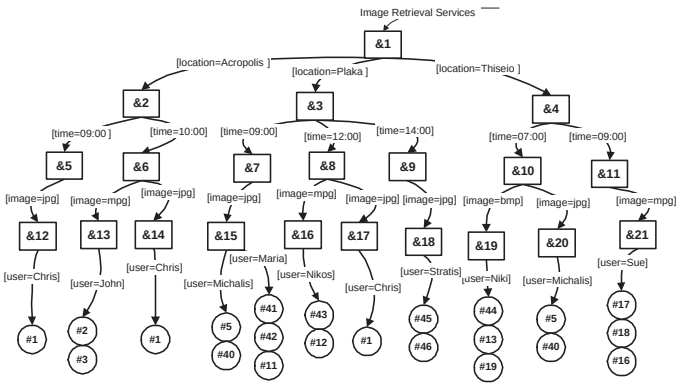

Figure 1: Part of a context-aware service directory with image retrieval services.

Service directories are cognizant of neighboring directories only, so global knowledge of the network topology is unavailable. This approach is robust and scalable, because it ensures system operation even during the failure of some nodes, and also allows dynamic addition or removal of CASDs. In this sense, CASDs form a P2P network and henceforth they will also be called *peers*. Note that, while in this paper neighbor proximity refers to geographical terms, the same approach can be extended to study semantic, content-based proximity, where neighboring peers are considered those with similar content.

Mobile users issue keyword-based service requests that are semantically disambiguated by means of a service taxonomy (i.e. a taxonomy of service categories, which is domain-specific and has a hierarchical structure) [10], thus leading to those service categories relevant to the user request. At the same time, a contextual query is formulated, describing the user's current situation, in order to be issued to the context-aware service directory. In this paper, our focus is on contextual query evaluation, after having found the relevant service categories.

3. CONTEXTUAL QUERY EVALUATION

In this section, we present the service discovery process assuming query workloads submitted to CASDs organized in a P2P setting. We use two main query evaluation approaches, namely local and distributed query evaluation, differentiated by the use or not of caching for performance enhancement and mitigation of the generated message overhead.

3.1 Distributed Query Evaluation

In the distributed query evaluation approach, each time a user issues a request for services, a contextual query is formulated representing her current context. This query is forwarded from the local CASD to its neighborhood, on every user request. Initially the query is submitted to the local CASD (henceforth also called *originator*), which performs the following tasks: a) forwards the query to its immediate neighbors, and b) evaluates the query locally. The neighboring CASDs follow the same procedure as long as the maximum hop count H is not reached. The matching results are sent back to the originator directly (not through the neighboring peers). The originator collects all results, and sends them back to the requesting device.

The hop count parameter H determines the range of the search, in terms of the number of steps that the query will be forwarded to a peer's neighborhood, starting from the originator. It is clear

that increasing values of H result in exponential explosion of the number of peers that receive and process the query - increasing thus the probability of reaching multiple and high quality answers to be reached. On the other hand a bulk of messages is exchanged due to the exponential nature of the setup.

3.2 Caching and Local Query Evaluation

Local query evaluation is a hybrid approach that employs query forwarding and caching of results in order to enhance the overall P2P network performance. Each peer (SD) stores service descriptions, both local and cached ones, as well as relevant information needed to ensure cache consistency. Three data structures are identified as essential parts of a peer's SD_i internal architecture: a) a context-aware service directory (CASD) represented as a graph, b) a cache used for services' descriptions storage, with similar structure to CASD, and c) a list L with the identifiers of the service directories SD_j ($j \neq i$) that currently cache copies of SD_i's services.

Consider the case of a mobile device u located in a cell controlled by a peer A and issuing a request Q for services. At the same time a contextual query Q_{cxt} is formulated incorporating the current context of the request (in terms of location, user profile, device capabilities, etc). Q_{cxt} is wrapped in a service request message (S_{req}) and then sent to the local service directory, say SD. SD supplements S_{req} with the hop count H and its peer id, forming thus a S_{reqf} message that is forwarded to all neighboring directories ($N(SD)$). Each directory receiving S_{reqf}, reduces H by one and further forwards it to its neighbors if $H > 0$. Moreover it evaluates Q_{cxt} and the matching services are returned to the originator A, via S_{res} messages, and at a next step to the requesting device u. Note that each S_{res} contains service descriptions retrieved both from the CASD and the cache of a peer. Finally, A merges the retrieved service descriptions with its current cache contents, for context Q_{cxt}, in order to serve further requests of the same context locally.

Fig. 2 presents a user mobility scenario and the respective message exchange when u moves from the cell controlled by peer A to the cell of peer B. At first, a message M_u (user moved) with the descriptions of the services $u.S$ offered by u, is sent from peer A to peer B. Peer B's CASD is now responsible for publishing $u.S$. This message is followed by M_l (list moved) that transfers from A to B the list of peers $A.L$ that cache parts of $u.S$ (in our case peers C and D). Cache consistency regarding $u.S$ is henceforth enforced by B, that sends I_c messages to C and D in order to update their cached copies of $u.S$. An I_c (invalidate cache) contains the id of the new peer that publishes $u.S$ and the context of u in the new cell. Assume that the context of $u.S$ is different in the new cell in terms of the location dimension (see Fig. 1), e.g. in cell B [location=Thisseio] while in cell A [location=Acropolis]. Moreover, peers C and D are interested only in services offered in the area of Acropolis. As a result, cache replacement is triggered and I_l (invalidate list) messages with their peer ids are sent from C and D to peer B in order to remove them from its list L.

3.3 Issues on Caching Query Results

Non-uniformly distributed query workloads motivate caching, in order to avoid repeated processing of successive, identical requests q_i. Assuming a discrete set of possible queries and a statistical distribution of the expected query workload, a *threshold* needs to be defined that signals caching initiation.

Cache consistency is usually maintained either by push-based or by pull-based approaches (as well as some variations). In the former approach, the data source is responsible for updating the content cached by clients, whereas in the latter approach, the clients must periodically ask for updates. Our approach is push-based and enforces cache consistency through cache updates whenever a change occurs, for example whenever a user moves to a different cell.

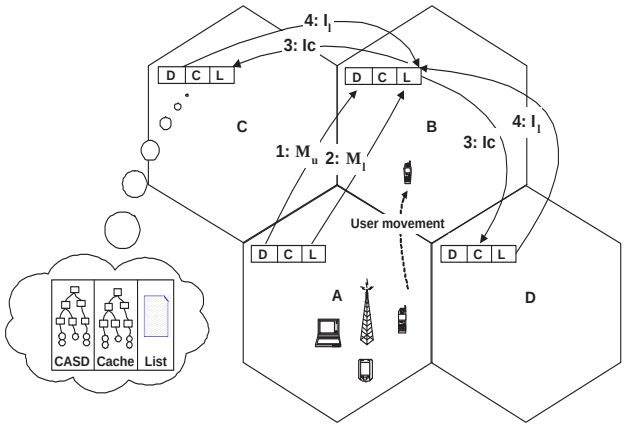

Figure 2: Message exchange resulting from user mobility

Cache forwarding refers to cached content propagation among peers, i.e. peer C requests from B services for context Q_{cxt} and peer B returns its local services, as well as cached services that belong to peer A. If cache forwarding is disabled, a peer's cache can not contain service descriptions beyond neighboring (defined by H) directories. On the other hand, when cache forwarding is enabled, a peer may eventually contain service descriptions from any service directory. However, this will result in caching large portions of service directories, frequent cache replacements and, consequently, performance degradation.

Cache replacement strategy is a critical aspect of any caching scheme and impacts overall performance. In a peer-to-peer setting relatively large hop counts may return increased number of results, thus causing frequent cache replacements. In this approach, we used least frequently used (LFU), but we intend to study the effect of other cache replacement schemes in our future work.

Mobility of data sources determines the efficiency of context-based caching. High mobility of services causes frequent cache updates, due to changes in the location of data sources, and consequently higher maintenance costs. On the other hand relatively low mobility favors the caching approach.

4. IMPLEMENTATION AND EXPERIMENTAL RESULTS

We implemented the proposed architecture of distributed service directories using the JXTA framework [13] for P2P applications. Each peer is represented as a JXTA peer that contains a list of neighboring peers. Neighbors are assigned to peers at startup to simulate coverage of a geographical region (each peer corresponds to a cell). Direct communication is achieved through exchange of messages between neighbors. Each peer contains a number of services for various contexts and a cache for storing locally service descriptions that belong to services outside its cell. The contextual service requests (Q_{cxt}), issued by mobile devices, are simulated by means of random query distributions in each peer. There are two functional modes: plain query forwarding or caching. Upon receiving a S_{req}, the peer forwards the query to its immediate neighbors and evaluates it locally. The originator peer retrieves the results

(sets of contextual paths) of the query and, if caching is enabled, caches them, in order to process further identical requests locally, in an autonomous way.

We performed extensive experiments on the implemented system. The basic experimental setup comprises $N = 25$ or 100 peers, with $n = 4$ neighbors per peer in both cases. Each peer was assigned $S = 1000$ services at startup, uniformly distributed under $C_s = 100$ different contexts (if we consider $D = 3$ contextual dimensions, having each 4 or 5 discrete values). Unless mentioned otherwise, the cache size is 20% of the directory size, the caching threshold t is set to 5 query occurrences and the hop count H is 1. We simulated query workloads by means of statistical distributions that show the frequency of occurrence for each of the 100 different contextual queries. A total of 2000 queries were generated in most of the cases. We allow cache forwarding in all experiments presented here.

The cache hit ratio (*CHR*) is the primary metric in this approach, because it defines the percentage of the queries that can be processed locally, without sending any messages to neighbors, thus influencing the overall system performance. Our goal is to measure the *CHR* and come up with design guidelines for P2P service directory design. We start recording *CHR* results after 80% of the peer cache has become full to eliminate the cache warm-up effect. In particular, we measure the *CHR* of each peer, for every 10 queries issued at it. We compute the average cache hit ratio of all participating peers and present comparative charts (see Fig. 3, 4) for varying:

1. cache sizes
2. query distributions

We also performed a set of experiments for varying caching threshold values, and found out that it does not influence the *CHR* significantly.

At first (see Fig. 3), we study the effect of different cache sizes on *CHR*. We tested different cache sizes ($c = 10\% - 100\%$), as a percentage of the directory size, for a Gaussian distribution (mean=50, st.deviation=15) of 100 distinct queries. Generally, increasing the cache size, results in *CHR* augmentation. However, after 650 queries and in order to increase the *CHR* from 68% to 80%, the cache size must increase from 10% to 100% of the directory size, so we educe that a relatively small benefit comes with a substantial cost for high *CHR*s. Also, notice that small cache sizes (10%) result in sufficiently good performance on the long run. Nevertheless, if a high *CHR* is required, then one should choose the largest cache size available.

Query distributions play an important role in the performance of caching. In Fig. 4, we compare the query workloads generated by

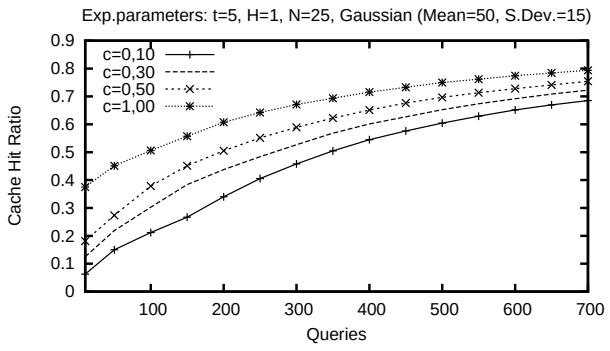

Figure 3: Effect of cache size on cache hit ratio.

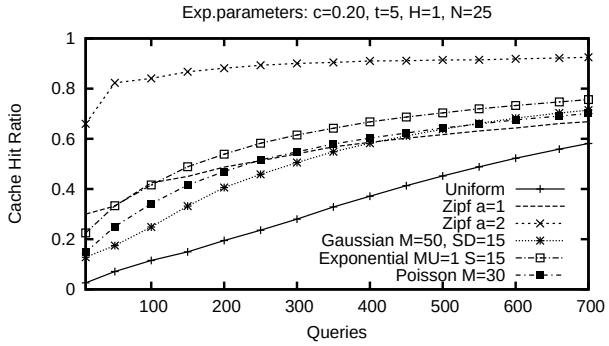

Figure 4: Effect of different query distributions on cache hit ratio.

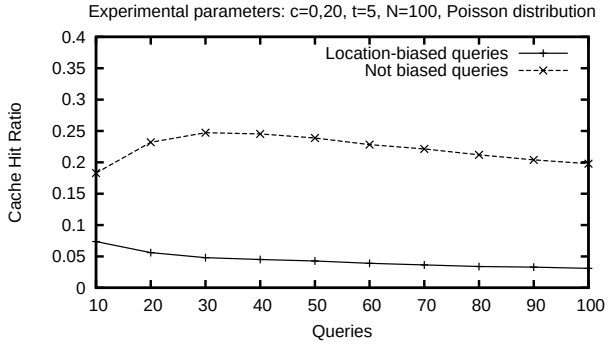

Figure 5: Cache hit ratio degradation vs query selectivity.

random distributions. Besides the uniform distribution, we chose three distributions for query workloads that can represent very frequent occurrences for some queries, as well as two variants of the zipfian distribution. As mentioned before, in our application scenarios, contextual queries tend to present high frequency at certain locations and times. The chart shows that the zipfian query distribution (with parameter $a = 2$) performed better, in terms of *CHR* (nearly 90%). As expected, the uniform distribution performed worse than the rest, however for the particular experimental setup the achieved *CHR* can be considered satisfactory. Generally, the more skewed the distributions, the higher *CHR* values they presented.

Then we studied the effect of query selectivity on *CHR*. The intuition is that for queries of low selectivity (i.e. many results returned) we expect that due to the high volume of the results frequent cache replacements take place. We conducted an experiment, shown in Fig. 5, where we used 100 peers to represent 4 neighboring geographic locations (25 peers per location) and each peer held services having as contextual value the corresponding peer's location. We performed two measurements for: a) location-biased queries for each peer and b) uniformly distributed queries, in order to have low and high query selectivity respectively. The results show that location-biased queries exhibited a very poor performance (small *CHR*), due to the low query selectivity. In other words, each query resulted in a large number of services, which was comparable to the cache size, thus resulting in very frequent cache replacements (practically the result of each query caused cache replacement). The lesson learned here is that our approach regarding context-based caching in a P2P architecture requires careful parameter tuning, especially in the case that cache size is comparable

to query selectivity.

5. RELATED WORK

The latest UDDI specification(ver.3.0) [12] addresses the challenge of service registry distribution, however its perspective is different compared to our approach (UDDI mainly focuses on avoiding service key collision in different registries).

Enhancing existing service discovery mechanisms, with respect to context, has been studied both in the WASP project [8] as well as in the CB-SeC framework [5]. The former approach supports semantic and contextual features, while the latter adopts an agent-oriented approach. Yet another approach [6] introduces the use of context attributes, as part of the service description.

There is an increasing interest in the research community about caching in P2P networks. Cache updates of mobile agents organized in a P2P manner is discussed in [7], whereas an approach regarding distributed caching in Gnutella-like networks, based on distributing the query results among neighboring peers, is presented in [11]. In [3], the authors propose building a P2P service directory by grouping together service entities semantically. Compared to the above, our approach deals with caching in P2P service directories, where caching of contextual service descriptions is motivated by query workload distribution.

6. CONCLUSIONS & FUTURE WORK

In this paper, we presented our approach regarding P2P service discovery based on caching and context-awareness and proved that taking into account context contributes to efficient service discovery. We claim that the nature of contextual requests favors caching, because these tend to present high frequency of occurrence at specific locations and times. The contribution of our work is the design of initial guidelines for service discovery in a P2P architecture of service directories. The conclusions of the experiments we conducted follow:

- The cache performance (*CHR*) increases with the size of the cache. On the other hand we noticed that after some point (around 68%*CHR*), we need an excessive cache size increase to achieve a relatively small performance gain.
- The caching threshold t does not play any significant role in *CHR* therefore it does not need to be taken into account as a design factor.
- Query distribution is a significant factor regarding cache efficiency. In general, skewed distributions achieve higher *CHR*s. Among the distributions we used in our experiments, caching proves to be mostly effective for the zipfian query distribution with parameter $a = 2$.
- Query selectivity: when query results volume becomes comparable to the cache size, then frequent cache replacements take place. As a result *CHR* substantially degrades. This needs to be taken into account when designing cache parameters and replacement schemes.

Our future work will focus on studying the effect of user mobility patterns on caching contextual service descriptions. Furthermore, we will continue our experiments towards studying performance in the case of skewed service distribution, where most users/services are located in a only few peers. Finally, extensions of the architecture will be considered, in order to handle semantic proximity among peers as well as other forms of contextual proximity besides geographical.

7. REFERENCES

[1] C. Doulkeridis, E. Valavanis, M. Vazirgiannis. *Towards a Context-Aware Service Directory*. In the Proceedings of the 4th VLDB Workshop on Technologies on E-Services (TES'03), pp. 54-65, Berlin, Germany, September, 2003.

[2] C. Doulkeridis, M. Vazirgiannis. *Querying and Updating a Context-Aware Service Directory in Mobile Environments*. In the Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence (WI'04), pp. 562-565, Beijing, China, September 2004.

[3] T.H. Hu, S. Ardon, A. Seneviratne. *Semantic-Laden Peer-to-Peer Service Directory*. In the Proceedings of the 4th IEEE International Conference on P2P Computing (P2P'04), Zurich, Switzerland, August 2004.

[4] C.S. Jensen. *Research Challenges in Location- Enabled M-Services*. In the Proceedings of the 3rd International Conference on Mobile Data Management (MDM'02), Singapore, January, 2002.

[5] S.K. Mostefaoui, B. Hirsbrunner. *Towards a Context-Based Service Composition Framework*. In Proceedings of the 1st International Conference on Web Services (ICWS'03), Las Vegas, Nevada, USA, June, 2003.

[6] C. Lee, S. Helal. *Context Attributes: An Approach to Enable Context-awareness for Service Discovery*. In the Proceedings of the 2003 Symposium on Applications and the Internet (SAINT'03), Orlando, FL, USA, January, 2003.

[7] E. Leontiadis, V.V. Dimakopoulos, E. Pitoura. *Cache Updates in a Peer-to-Peer Network of Mobile Agents*. In the Proceedings of the 4th IEEE International Conference on P2P Computing (P2P'04), Zurich, Switzerland, August 2004.

[8] S. Pokraev, J. Koolwaaij, M. Wibbels. *Extending UDDI with Context-Aware Features Based on Semantic Service Descriptions*. In the Proceedings of the 1st International Conference on Web Services (ICWS'03), Las Vegas, USA, June, 2003.

[9] Y. Stavrakas and M. Gergatsoulis. *Multidimensional Semistructured Data: Representing Context-dependent Information on the Web*. In Proceedings of the 14th International Conference on Advanced Information Systems Engineering (CAISE'02), Toronto, Canada, May 2002.

[10] E. Valavanis, C. Ververidis, M. Vazirgiannis, G. C. Polyzos, K. Norvag. *MobiShare: Sharing Context-Dependent Data and Services from Mobile Sources*. In Proceedings of IEEE/WIC International Conference on Web Intelligence (WI'03), Halifax, Canada, October, 2003.

[11] C. Wang, L. Xiao, Y. Liu, P. Zheng. *Distributed Caching and Adaptive Search in Multilayer P2P Networks*. In the Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04), Tokyo, Japan, March 2004.

[12] *The UDDI specification version 3.0* http://www.uddi.org

[13] *Project JXTA: A Technology Overview* http://www.jxta.org/