QUERY ROUTING IN NOMADIC ENVIRONMENTS WITH PUMAS

Angela Carrillo-Ramos[2], Marlène Villanova-Oliver[1], Jérôme Gensel[1]
Hervé Martin[1] and Miguel Torres[2]
[1]*LIG Laboratory, STEAMER Team, 681 rue de la Passerelle, 38402 Saint Martin D'Hères, France*
[2] *Pontificia Universidad Javeriana, ISTAR Team, Carrera 7 No. 40 – 62, Bogotá, Colombia*

Keywords: P2P systems, Agents, Query Routing, nomadic environment, adaptation.

Abstract: *PUMAS* is a framework based on agents and a *Peer to Peer* (*P2P*) approach, which allows nomadic users to access several information sources through different types of devices (eventually mobile). *PUMAS* provide the nomadic user with information adapted to her/his *preferences* and to the characteristics of the *context of use* (*i.e.* location, connection moment, inherent characteristics of the access device). In order to deliver adapted results, *PUMAS* relies on mechanisms for enriching the initial query, by means of adding criteria based on *user preferences* and *context of use*. This phase of query enrichment precedes a *query routing process* composed of three activities: the query analysis, the selection of sources that are able to answer to the query and the redirection of the query towards these sources. We present the algorithms and the knowledge managed by *PUMAS* agents in these activities and we illustrate the *query routing process* that *PUMAS* executes in order to deliver adapted information.

1 INTRODUCTION

Mobile Devices (*MD*) such as *PDA*, phones, laptops, *etc.*, are used to access distant information sources, and are also used as information sources by themselves. In this context, to guarantee access to several information sources and to share these resources among users, architectures and technologies with high communicative potential are required. This potential is one of the main characteristics of *Peer to Peer* (*P2P*) systems.

The handling of queries that come from nomadic users becomes an increasingly complex process since the data that can satisfy the query can be held by different information sources. Additionally to the number of potential sources, the heterogeneity of the source's structure and access methods can be an issue for their exploitation. Finally, it is difficult to provide the user with the best-adapted information to his needs and to the characteristics of her access device. These issues give rise to the need to previously analyze the queries, in order to identify the sources that will give the most appropriate answers to the query (this supposes that knowledge about the information managed by each source is available), then to evaluate and finally to integrate the results. These activities correspond to a traditional *query routing process* (Xu *et al*., 1999). However, these activities do not consider explicitly the adaptation of information. Our approach addresses this issue and proposes a query enrichment process, which comes upstream from the routing process. This enrichment "*augments*" the initial query by adding criteria that consider both *user preferences* and *context of use* of the current session. In our work, this *context* is composed of information about the features of the *MD*, the location, the access rights and the activities of the user (Carrillo, 2007).

This proposition of query enrichment is based on several factors that can influence the routing (in the sense that the sources are the addressees of the queries and that they can change): *i)* Changes of the user's location can produce changes in terms of access and information needs (Thilliez *et al*., 2004). We believe that this aspect is also sometimes valid when there is a device change; *ii)* User preferences are context-aware (Carrillo, 2007). When the query is submitted, all context changes involve potential consequences on the query, and consequently on the sources. *iii)* The technical constraints of the *MD* of a nomadic user can give rise to problems of information display, which are difficult to anticipate.

Carrillo-Ramos A., Villanova-Oliver M., Gensel J., Martin H. and Torres M. (2008).
QUERY ROUTING IN NOMADIC ENVIRONMENTS WITH PUMAS.
In *Proceedings of the Fourth International Conference on Web Information Systems and Technologies*, pages 125-130
DOI: 10.5220/0001513401250130

In order to considerate these factors, we propose *PUMAS* (*Peer Ubiquitous Multi-Agent System*), an agent based framework that aims at providing a nomadic user with adapted information according to her *preferences* and to the *context of use*. *PUMAS* also offers means for questioning several sources, which correspond to *Information Systems* (*IS*) executed on servers, or simple files stored on other *MD*. *PUMAS* agents perform the enrichment mechanism of the initial query by adding criteria taking into account this way both *user preferences* and *context of use*. This phase of query enrichment leads to a *routing process*.

In *PUMAS*, users communicate with each other by means of a *hybrid P2P* architecture. We have followed the recommendations of (Shizuka *et al.*, 2004) which consider that the use of *Hybrid P2P* architectures is appropriated for: *i)* the prevention of security problems related to agent mobility; *ii)* point to point or broadcast communication among agents; *iii)* management of agent state and provided services (*e.g.*, "*connected*", "*disconnected*", *etc.*). The main characteristics of the *hybrid P2P systems* in *PUMAS* are (Carrillo *et al.*, 2005): *i)* a *MD* can communicate with a specific *IS* by giving to the *Router Agent* (*RA*) the *ID* of the required *IS* as a parameter in the query. The *RA* then transmits the query to this *IS* (communication from agent to agent).; *ii)* an agent is autonomous to connect/disconnect from *PUMAS* whenever the agent requires it.

PUMAS is composed of four *Multi-Agent Systems* (*MAS*) in charge of connection, communication, information sources management and information adaptation. In a general way, the *MAS* are similar to *P2P systems* in several points: each agent is a peer in the sense that it can perform its own tasks independently of the server and of the other agents. *P2P systems* (Röhm *et al.*, 2000) are characterized by: *i)* direct communication among peers without communication through a specific server; *ii)* the autonomy of a peer for performing its assigned tasks. Considering a *P2P* approach, a *MAS* must represent the required knowledge for each agent to perform the different roles that it can assume (*e.g.*, client, server) (Panti *et al.*, 2002).

This paper is organized as follows: Section 2 defines the *query routing process* in *P2P* systems. Section 3 depicts a brief *PUMAS* framework overview and is dedicated to the *query routing process* in *PUMAS*. In particular, we present the algorithms and the knowledge managed in the activities of this process: the query analysis and its redirection to the sources able to answer it. Finally, we present the conclusion and future work.

2 QUERY ROUTING PROCESS IN P2P SYSTEMS

(Xu *et al.*, 1999) define *query routing* as a general problem which is based on two main activities: *i) Query Evaluation* using the most relevant sources, and *ii) Result integration* of the data generated by different sources. In order to perform these two activities, several issues must be considered: *i) Source selection,* which consists in the analysis of the user's query in order to determine the sources that are able to fulfill the query. In order to resolve this problem, the system must know the kind and structure of the information managed by each source; *ii) Query Evaluation* which is performed by the sources selected at previous step. Since there can be heterogeneous data representations among different sources, the original query must be translated into terms that the questioned sources can understand; *iii) Result Integration*: since results can be returned by different sources, they must be integrated into a single one that will be returned to the user.

In this paper, we focus on the first identified problem (*Source Selection*). We have classified the possible solutions to this problem into three main categories: *i)* solutions that use a *gathering* of peers for query answering, *ii)* those based on *filters* and data *matching* between terms of a query and the data of a source and, *iii)* those based on *Trust and Reputation* strategies.

Considering the *gathering of* peers that are able to answer to the query, (Brunkhorst *et al.*, 2003) propose to specify the peers associated to a super-peer and the information managed by each peer within the context of the *P2P system* based on schemas. The super-peers form a small subset of peers in which each peer has specific responsibilities and the capability of gathering with others in order to perform tasks such as query routing, mediation in conflict resolution and teamwork. These authors have defined a set of indexes necessary to perform the *query routing* among the (super-)peers of the system. (Kokkinidis *et al.*, 2006) propose to gather peers that share the same information schemas in order to define plans for query answering in a distributed way. In this proposal, the super-peers are responsible for the *query routing*, and the peers are responsible for handling the queries.

(Koloniari *et al.,* 2004) define *query routing* as a mechanism for determining the location of nodes containing documents, which can fulfill the query. This location is based on a mechanism of document and filter *matching*. The *filters* are specialized data

structures that gather general information of large collections of documents, and which redirect the queries only towards the nodes that contain relevant information. Each node has two types of filters: the *local filter,* which gathers general information of documents that are stored locally in the node, and the *merged filters,* which gather general information of documents of the neighbor nodes. When a query arrives to a node, this node verifies its *local filter* and uses the *merged filters* in order to redirect the query only towards the nodes whose filters correspond to the query. (Xu *et al.*, 1999) present some strategies based on techniques of *clustering* for the sources' selection. These strategies are achieved in two steps: *i)* construction of knowledge about the sources that compose each cluster and about information managed by each one. *ii)* Ranking of sources: the position of each source in the ranking is determined considering the total of the matching scores obtained by the clusters to which the source belongs and the clusters' size.

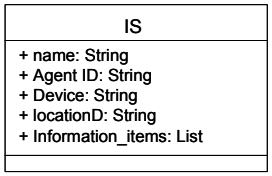
In order to optimize the *query routing process*, (Agostini *et al*., 2004) propose to use several metrics related to: the reliability of the sources, their capabilities to satisfy the user's information needs and their reliability for delivering information to users. Agostini *et al.* propose a strategy of *Trust and Reputation* allowing the selection of peers that are able to answer the query. The *trust* assigned to an agent is the degree of security estimated regarding the quality of its answers. *Trust* is based on its *reputation*. The *reputation* of an agent is the opinion that other agents have about it; this opinion is obtained considering the results provided by an agent to previous inquiries and additional criteria such as the time taken to answer, the exhaustiveness of the answers, the relevance of the answers, *etc*. *Reputation* is a value in a qualitative or quantitative scale calculated by each agent. A system of peer *reputation* compiles, distributes and manages information about previous behaviors of peers. The strategy of *Trust and Reputation* proposed by Agostini *et al.* is based on the following process: a component named "*seeker*" selects the peers that are able to answer to the query *Q* with the highest probability considering established criteria (*e.g.* first to answer, quickest to answer, most reliable answer, most trusted peer, etc.). In order to decide, the *seeker* creates and manages a list $<p_1, p_2,...p_k>$ of trusted peers for sending the query. The list is sorted using a decreasing trust level. The *seeker*'s strategy to answer to a query is as follows: first, the *seeker* asks p_1, then p_2, and so on, until it receives relevant answers based on the established criteria. It is important to notice that the list of trusted peers can evolve in time.

In a nomadic environment, information required by a user can evolve in function of the *context of use*. For example, a user can ask for a list of restaurants and for getting those located in the street where she is and which are open at the time she formulates the query. The techniques of *query routing* presented previously do not allow the management of this kind of context evolution. The following section presents *PUMAS*, a framework, which considers these aspects by means of enrichment mechanisms of the initial query in order to adapt information to the user.

3 QUERY ROUTING IN PUMAS

During an information search process in nomadic environments, a user can be confronted with several problems. One can mention: *i)* access problems related to the characteristics of networks and of the user's *MD*; *ii)* a lack of adaptation of the results considering user's characteristics and preferences, and technical constraints of the user's *MD*; *iii)* a lack of mechanisms for searching distributed information on several sources and devices (servers or *MD*). In order to resolve these problems, we have proposed *PUMAS*, a framework based on agents, which provides nomadic users with information adapted to their characteristics and those of their *MD*. *PUMAS* also provides means of interrogating several sources (*e.g. Information Systems*, *IS*). In the remainder of this paper, we focus on the querying of sources of *IS* type. The *PUMAS'* architecture is composed of four *MAS* (their complete description are given in (Carrillo, 2007)): *i)* The *Connection MAS* provides mechanisms for facilitating the connection of different types of *MD* to different *IS*. *ii)* The *Communication MAS* assures transparent communication between the users' *MD* and the system, and applies a *display filter* (with the help of *Adaptation MAS* agents) in order to present information in an adapted way, considering the technical constraints of the *MD*. *iii)* The *Information MAS* receives queries from users, redirects them to the *IS* that are able to answer them, applies a *content filter* (with the help of *Adaptation MAS* agents), considering the user profile, and returns the results to the *Communication MAS*. *iv)* The *Adaptation MAS* communicates with agents of the three other *MAS* in order to exchange information about user, connection, *MD* and communication features, *etc*.

In this paper, we focus on the algorithms and the knowledge managed for the *query routing process* in *PUMAS*, achieved by the *Router Agents* (*RA*) which receive enriched queries. These activities are inspired of the work of (Xu *et al.*, 1999). When a *RA* receives a query, this agent can send it to a specific or several *ISAgents* (belonging to the *Information MAS*), or it can split the query in sub-queries which are sent to one or several *ISAgents*.



(assert (IS (name PharmacyIS) (agentID PharmacyISA) (device server) (locationD "North Hospital") (information_items "patient medicines" "dose" "medicines")))

Figure 1: IS Representation: Class and Fact.

The ***first activity*** of the *query routing process* is the *query analysis*. In this activity, the *RA* analyzes the complexity of the query and classifies it in "*simple*" (if it can be processed by only one *ISAgent*) or "*complex"* (if several *ISAgents* are required to process it). In this case, this activity will split the query in sub-queries. This analysis is based on the facts related to the *IS*, stored in a knowledge base managed by the *RA*. A fact is a knowledge piece which describes a real world element. In order to clarify its definition, we also show its representation as a *UML* classe (see Figure 1). The *RA* also analyzes the adaptation criteria of a query (*e.g.*, location, user activities), the list of the addressees of a query, *etc.* After this analysis, the *RA* decides if it must split the query in sub-queries. Let us suppose for example that a doctor wants to consult the results of *the medical analysis of a patient, her diet and her prescribed medicines*. We suppose that no *IS* can completely answer this query. This query is identified as "*complex"*, and is therefore split by the *RA* in three sub-queries, concerning respectively "*medical analysis*", "*diet*" and "*prescribed medicines*". Such a division is based on the stored knowledge obtained by the *RA* regarding the managed information by the different *IS* as a *JESS* fact (see Figure 1).

In order to split the query, the *RA* identifies the query items [$item_1$, $item_2$... $item_n$]. In our example, $item_1$ corresponds to "*medical analysis*", $item_2$ to "*diet*" and $item_3$ to "*prescribed medicines*". Then, this agent searches for the *IS* which manage an equivalent item. Two items are "*equivalent*" if they are semantically equals (*i.e.*, if both items have the same meaning) or are equal syntactically (*e.g.*, equality in strings of characters). The applications' designers define the *equivalence relationship*. They also implement our *Matching Algorithm*. This algorithm generates an *Information System List* (*ISL*) containing tuples (*IS, <items managed by the IS>*):

```
(1) Initialize ISL answering particularly to the query (ISL)
(2) nIS← 0 ;
(3) i← 1 ; // index of the items of the query
(4) While i ∈ [1, n] do // index of the items of the query
(5)         j← 1 ; // index of the IS
(6)         While j ∈ [1, s] do // index of the IS known by the RA
(7)             m← 0 ; // index of the information items of ISj
(8)             While m < size (list of info items managed by ISj) do
(9)                 If Compare (itemi,info_itemm managed by ISj) then
(10)                    nIS ← nIS +1 ;
(11)                    ISL← ISL ⊕ (ISj, <itemi>) ;
(12)            End If
(13)            m ← m + 1 ;
(14)        End While
(15)        j ← j + 1 ;
(16)    End While
(17)    i ← i + 1 ;
(18) End While
(19) If nIS is equals to 0 then
(20)    query_type ← "without answer"
(21) else
(22)    sid← countDifferent (ISL) ;
(23)    If sid is equals to 1 then
(24)        query_type ← "simple"
(25)    else // sid is upper to 1
(26)        query_type ← "complex"
(27)    End If
(28) End If
(29) ISL ← Compact(ISL)
```

First, the *ISL* is empty (see line (1)). Then, the variable *nIS* (containing the number of *IS* managing the equivalent items to those of the query, see line (2)), is initialized. For each item of the query ($item_i$ with $i \in [1, n]$, lines (3) to (18)), it searches the IS_j (IS_j with $j \in [1, s]$) which manage information items equivalents to the analyzed one. The "*Compare*" method verifies the equivalence of the items (see line (9)). If they are equivalent, the algorithm increases the variable *nIS*, and adds into the *ISL* a tuple whose first term corresponds to an IS_j and the second one corresponds to the analyzed item (see lines (10) and (11)). When all the query items have been analyzed, the algorithm verifies the value of *nIS* (see lines (19) to (27)). If this value is zero, it means that there is no *IS* able to manage equivalent items from the query. In this case, the query is tagged as "*simple*". In the other case, the algorithm analyzes the *ISL* in order to know the number of different *IS* which manage items equivalent to those of the query, in order to characterize the query as "*complex*". This number is calculated using the

"*countDifferent*" method (see line (22)). Finally, the algorithm uses the "*Compact*" method which leaves in the *ISL* only one tuple associated for each *IS*. The second term of this tuple corresponds to all the query items managed by the *IS*. We illustrate the "*countDifferent*" and "*Compact*" methods in order to clarify them: Let us suppose that the *RA* has identified items: i_1, i_2, i_3, i_4 and i_5, and it knows the following *IS*: IS_1, IS_2, IS_3, IS_4, IS_5 and IS_6. After the items' comparison (see lines (4) to (19)), we suppose that the *ISL* is composed of the following tuples:

$(IS_1, <i_1>)$ $(IS_3, <i_1>)$ $(IS_5, <i_1>)$ $(IS_1, <i_2>)$ $(IS_3, <i_2>)$
$(IS_4, <i_2>)$ $(IS_5, <i_2>)$ $(IS_1, <i_3>)$ $(IS_3, <i_3>)$ $(IS_4, <i_4>)$
$(IS_5, <i_4>)$ $(IS_1, <i_5>)$ $(IS_3, <i_5>)$ $(IS_4, <i_5>)$ $(IS_5, <i_5>)$

The "*countDifferent*" method will return the value 4 because only IS_1, IS_3, IS_4 and IS_5 appear in the tuples. The "*Compact*" method leaves only one tuple for each *IS*. That is:

(IS1, <i1, i2, i3, i5>) (IS3, <i1, i2, i3, i5 >)
(IS4, <i2, i4, i5 >) (IS5, <i1, i2, i4, i5>)

Following the algorithm, we can conclude that the query is "*complex*" and the *ISL* is composed of:

(PharmacyIS, <"prescribed medicines">)
(NutritionistIS, <"diet">)
(MedicalAnalysisLaboratory, <"medical analysis">)

An item of the query can be managed by several *IS*, then it is necessary to select the most appropriated *IS* for answering them. The ***second activity*** corresponds to *the selection of the IS*. In this activity, a query can be rerouted towards a specific agent or towards a group of agents. If the addressees of a query are known, the selection is relatively easy. Otherwise, the *RA* selects the *IS* and composes the network of neighbors (considering the *ISL* produced during the previous activity). This process is based on the ideas of (Yang *et al.*, 2004) who propose an information retrieval process in *non structured P2P systems*, where each node has a data collection shared with other nodes. When a user submits a query, her corresponding node becomes the sender of the query and it can send messages (including the query) to several of its neighbors. When a neighbor receives the query, it handles the query using first its local information. If the node finds some results, it returns them to the sender node. In our proposal, a *peer* is named "*neighbor*" of other peers, if it satisfies a set of characteristics (criteria defined in the *user preferences*) such as a close location, similar activities, roles, knowledge, colleagues working in the same group, *etc*. However, characteristics are not restricted to proximity criteria.

Since several agents can answer to the same query, the *network of neighbors* can be composed of these agents. This gathering is useful when the *RA* does not have any information about the *IS* or when it is the first time that this agent works with the neighbors. In order to avoid useless, redundant or unusable communications, and to select the most relevant neighbors, the *RA* applies query's adaptation criteria. For example, if the criterion is location, the network is composed of the close neighbors; if the user queries depend on his previous queries, the *RA* must redirect them to the trusted neighbors; if there is no criterion defined, the *RA* analyzes the trust level of its neighbors. The *RA* associates a trust level to each neighbor, based on previous answers applying the *Trust and Reputation* strategy proposed by (Agostini *et al.*, 2004) (see section 2). If a query has been split in several sub-queries in the *analysis activity*, the *RA* analyzes the agents that can answer to each sub-query and these agents compose the *network*. For each sub-query, it is necessary to select the *IS* that are able to answer. Finally, the *network* is composed of the aggregation of the different generated sub-networks for each sub-query. The *RA* considers the *ISL* produced in the *analysis activity*. In our example, this agent selects the *PharmacyIS*, the one of the *Medical Analysis Laboratory* and the one of the *Nutritionists*. These *IS* are the only ones able to answer to each one of the three sub-queries. These sub-queries are sent to the corresponding *ISAgents*. We give below the example of the sub-query $Query_1$ produced by the *IS* of the *medical analysis laboratory* where the *ISAgent* is named *MedicalAnalysisLaboratoryISA* (see Figure 2)). According to an equivalent principle, $Query_2$ and $Query_3$ will be sent respectively to the agents *NutritionistISA* of the *NutritionistIS* and the *PharmacyISA* of the *PharmacyIS*.

(*assert* (*Query* (*QueryID Query1*)
(*UserID "Doctor John Smith"*)
(*ISA MedicalAnalysisLaboratoryISA"*)
(*IS "MedicalAnalysisLaboratoryIS"*)
(*required_info "Medical analysis"*)
(*parameters "patient name" "date"*)))

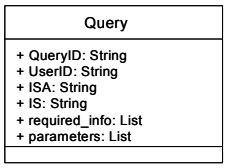


Figure 2: Query Representation: Fact and Class.

The ***third activity*** is *the redirection of queries to the IS*. In this activity, once the *RA* has identified the potential *IS* (neighbors), it must analyze the trust level associated to each one of them, to determine a query's redirection protocol. This information about trust level can be unavailable, if it is the first time that the *RA* executes this query or that it works with these *IS*. In the same way, the trust levels of the neighbors can be similar. In these conditions

(absence of established trust), the *RA* sends the query in "*broadcast*". When the *RA* has information exploitable in terms of trust, it redirects the query in a sequential way, beginning by the most trusted agent. The answer to the query will be the one generated by the first agent that fulfills it. If the *RA* does not receive any answer, the user will be informed about it. If the *RA* only knows the neighbors able to answer the sub-queries ($query_1$, …, $query_N$), the *RA* sends directly the sub-queries to these neighbors. In our example, the *RA* sends $Query_1$ to the *MedicalAnalysisLaboratoryISA*, $Query_2$ to the *NutritionistISA* and $Query_3$ to the *PharmacyISA*. The *RA* must collect and classify the obtained answers from the different agents and select the most relevant ones considering established adaptation criteria. The scenarios that present the sending and enrichment of query and the reception of results can be found in (Carrillo *et al.*, 2005).

4 CONCLUSIONS

When a user formulates queries, the results can come from different *Information Sources* (*IS*). In this paper, we have defined a *query routing process* as a mechanism which analyzes the query, and performs a matching (semantic or syntactic) between query's items and *IS*' items. This matching is achieved in order to select the *IS* able to answer to user queries. After identifying items and the recognition of the *IS* that are able to manage equivalent items, this process splits the query. A *Router Agent* considers adaptation criteria provided by the user (*e.g.* location, user's activities, preferences) in order to choose the most appropriated *IS* for answering the query. Finally, this process must collect and classify the query results. We have illustrated the *query routing process* by means of an example implying the different *IS* of a hospital in which a doctor asks for information about the *prescribed medicines*, the *medical analysis* and the *diet* of a patient. Our future work aims the definition of an algorithm for the collection and analysis of results coming from one or several *IS*.

REFERENCES

Agostini, A., Moro, G., 2004. Identification of Communities of Peers by Trust and Reputation. In *AIMSA 2004, 11th Int. Conf. on Artificial Intelligence: Methodology, Systems, and Applications*. Springer, p. 85-95.Brunkhorst, I., Dhraief, H., Kemper, A., Nedjl W., Wiesner, C., 2003. Distributed Queries and Query Optimization in Schema-Based P2P Systems. In *DBISP2P, 1st Int. Workshop on Databases, Inf. Systems, and P2P Computing*. Springer, p. 184-199.

Carrillo-Ramos, A., Gensel, J., Villanova-Oliver, M., Martin, H., 2005. A Peer Ubiquitous Multi-Agent Framework for providing nomadic users with adapted information. *In AP2PC 2005, 4th Int. Workshop on Agents and P2P Computing*. Springer, p. 159-172.

Carrillo-Ramos, A., 2007. Agents ubiquitaires pour un accès adapté aux systèmes of information : Le Framework PUMAS. *Doctoral Thesis of the University Joseph Fourier, Grenoble, France*. (March 5th, 2007)

Kokkinidis, G., Lefteris, S., Christophides, V., 2006. Query Processing in RDF/S-Based P2P Database Systems. *Semantic Web and P2P*. Springer, p. 59-87.

Koloniari, G., Pitoura, E., 2004. Content-Based Routing of Path Queries in Peer-to-Peer Systems. In *EDBT 2004, 9th Int. Conf. on Extending DB Technology*. Springer, p. 29-47.

Panti, M., Penserini, L., Spalazzi, L., 2002. A Multi-Agent System based on the P2P model to Information Integration. In *AAMAS 2002, 1st Int. Conf. on Autonomous Agents and Multi-Agent Systems*. http://www.agentcities.org/EUNET/Projects/acnet_proj_38.pdf (Last Access: September 2006)

Röhm, U., Böhm, K., Schek, H., 2000. OLAP Query Routing and Physical Design in a Database Cluster. In *EDBT 2000, 7th Conference on Extending Database Technology*. Springer, p. 254-268.

Shizuka, M., MA, J., Lee, J., Miyoshi, Y., Takata, K., 2004. A P2P Ubiquitous System for Testing Network Programs. In *EUC 2004, Embedded and Ubiquitous Computing*. Springer, p. 1004-1013.

Thilliez M., Delot T., 2004. Evaluating Location Dependent Queries Using ISLANDS. In *ISSADS 2004, Symposium on Advanced Distributed Systems*. Springer, p. 126-136.

Xu, J., Lim, E., Ng, W.K., 1999. Cluster-Based Database Selection Techniques for Routing Bibliographic Queries. In *DEXA 99, 10th Int. Conf. on Database and Expert Systems Applications*. Springer, p.100-109.

Yang, D., Xu, L., Cai, W., Zhou, S., Zhou, A., 2004. Efficient Query Routing for XML Documents Retrieval in Unstructured Peer to Peer Networks. In *APWeb 2004*. Springer, p. 217-22