Noname manuscript No.
(will be inserted by the editor)

Multi-scale query processing in vehicular networks

T. Delot · S. Ilarri · M. Thilliez · G. Vargas-Solar · S. Lecomte

Received: date / Accepted: date

Abstract In the last decade, a number of wireless and small-sized devices (e.g., PDAs, smartphones, sensors, laptops, etc.) with increasing computing capabilities have appeared in the market at very affordable costs. These devices have started to be embedded in modern cars in the form of on-board computers, GPS navigators, or even multimedia centers. Thus, the vehicles can carry useful information, acting as data sources for other vehicles. Recently, some works have addressed the problem of processing queries in such highly dynamic vehicular networks in order to share information between drivers. The proposed query processing techniques usually rely on a push model. Hence, each vehicle receives data from its neighbors and decides whether they are relevant enough to be stored in a local data cache. Then, the data may be used by a query processor to retrieve relevant data for the driver.

In this paper, we look at the problem from a broader perspective and discuss the interest of multi-scale query processing techniques in such context. The goal of such techniques is to exploit, at the mobile device's level, different access modes (e.g., push, pull) and various data sources (e.g., data cached locally, data stored by vehicles nearby, remote Web services, etc.) to provide the users with results for their queries. We highlight the most important challenges and outline some possible approaches. We also present a prototype of a first query evaluator developed using the Microsoft LINQ API.

Keywords Mobile Data Management · Vehicular Networks · Multi-Scale Query Processing · Network applications & services

1 Introduction

Mobile devices and wireless networks are now widespread, and continuous technological advances are observed in this area. Today, many people use small devices to retrieve different types of data every day. The importance of mobile data will grow even more in the upcoming years. Thus, ABI Research estimates that "In 2014, the volume of mobile data sent and received every month by users around the world will exceed by a significant amount the total data traffic for all of 2008"[1]. All these trends are motivating a great amount of research to try to develop suitable data management strategies for mobile computing.

Research in data management for mobile computing started already in the 90's, with several papers studying the problems introduced by the –at that time– new environment [6,20,21,37]. Specifically, the emergence of both handheld devices and wireless networks has had a strong impact on query processing techniques [22,

T. Delot, M. Thilliez and S. Lecomte
University Lille North of France
LAMIH FRE UHVC/CNRS 3304
Valenciennes, France
Tel.: +33 3 27 51 19 56
Fax: +33 3 27 51 19 40
E-mail: Firstname.Lastname@univ-valenciennes.fr

S. Ilarri
University of Zaragoza
IIS Department
Zaragoza, Spain
E-mail: silarri@unizar.es

G. Vargas-Solar
CNRS, LIG-LAFMIA
Grenoble, France
E-mail: Genoveva.Vargas-Solar@imag.fr

[1] See `http://www.abiresearch.com/press/1466-In+2014+Monthly+Mobile+Data+Traffic+Will+Exceed+2008+Total`, Accessed May 9, 2011.

43]. Thus, mobile devices are constrained by limited resources in terms of autonomy, memory, and storage capacity. They also provide an intermittent connectivity subject to frequent disconnections. Moreover, the mobility of clients and/or data sources has highlighted the importance of supporting spatio-temporal criteria for data access and develop solutions which are context-aware [5,13]. Different types of queries are thus frequently considered in mobile environments [19], such as:

– Location Dependent Queries (LDQ) [19,38], which are queries whose answers depend on the locations of certain moving objects.
– Spatiotemporal Queries (STQ) [24], which include all queries that combine space and time and generally deal with moving objects.
– Continuous Queries (CQ) [3], whose answer is automatically refreshed as needed, in order to support the frequent changes/updates in the query results (mainly due to the mobility of the relevant objects).

Besides, the notion of *query result* has also evolved due to the increasing mobility and dynamism of mobile computing contexts. Indeed, trying to obtain a complete query result, as traditionally considered in classical (distributed) databases, generally makes no sense here. Thus, for example, since the devices and the data sources can appear or disappear at anytime, it is unrealistic to assume a global schema maintaining the location of all the data sources. It is so completely impossible to determine whether all possible solutions have been retrieved in the result presented to the user or not. Therefore, the traditional *closed world assumption* is replaced by an *open world assumption* [8]. The objective of query processing techniques here is to retrieve the maximum number of results interesting for the user with respect to one or more criteria (e.g., result computation time, energy spent, financial cost, etc.).

Today the situation is still evolving and new elements play a key role [43]. Among them, we can highlight the availability of multiple data sources, multiple forms of network access, and the emergence of mobile P2P schemes. All these aspects have an important impact on the development of appropriate query processing techniques:

– On the one hand, when a query is submitted it is highly probable that *multiple data sources* will need to be accessed in order to compose the final answer, including: local data stored on the mobile device, data stored on other devices nearby, remote databases, Web services, or even sensors or the Internet. We define *multi-scale query processing* as any query processing that may need to access data sources of different types to obtain the answer to a query, generalizing the definition provided in [9].
  Every data source will have different features in terms of access latency, economic cost, reliability, data quality or accuracy, etc. In some cases, it may also happen that some data could be available in more than one source. The heterogeneity and the distributivity of the resources bring several challenges: 1) the relevant data sources must be selected by considering the requirements of the user query, the availability of data, and the special features of each data source; 2) the query has to be decomposed into several local queries that are sent to the different data sources for processing; and 3) the data retrieved must be correlated to assemble the final answer.
– On the other hand, there are now *multiple network connectivity options*. Thus, a mobile device can connect to the network by using 3G, Wi-Fi, Bluetooth, UWB, etc. All these communication options have different features in terms of bandwidth, access latency, battery consumption, economic cost, availability, reliability, etc. Thus, it is a challenge to select the best communication option at a certain moment according to predefined user preferences.
– Finally, it is not only possible to access data through a fixed network infrastructure (like in the classical mobile computing environment, considered in works such as [37]) but also through direct interactions in an *ad hoc mobile P2P network*. Thus, a device can retrieve or receive data diffused by other devices located nearby, thanks to the use of short-range wireless communications such as Bluetooth, Wi-Fi, or UWB.

Therefore, new data management and query processing techniques are needed. The techniques proposed should take into account the elements mentioned above, among others, in order to decide appropriate query execution plans and retrieve the results. In this paper, we will focus on the specific case of *vehicular networks* [30, 35], where vehicles can exchange data among themselves to inform drivers about interesting events (e.g., available parking spaces, accidents, traffic congestions, etc.) [10,44]. In this highly-dynamic scenario, data management for mobile P2P is at the same time promising and especially challenging. So, the purpose of this paper is to analyze the challenges that arise for query processing in vehicular networks, with an emphasis on *multi-scale query processing*, and outline possible solutions.

The structure of the rest of the paper is as follows. In Section 2, we briefly describe the context of vehicular networks and the motivation to develop this work.

In Section 3, we describe different modes of data access for vehicular networks: access to remote services, push-based approaches, and pull-based approaches. Section 4 introduces multi-scale mobile query processing and describes the schema used to process queries. In Section 5, we explain how to optimize multi-scale mobile queries. In Section 6, we describe a first prototype that we have developed, using the Microsoft LINQ API, to illustrate some of our proposals. Section 7 presents some related works. Finally, in Section 8, we summarize our conclusions and outline some lines of prospective research.

2 Context and Motivation

A *vehicular ad-hoc network* (*VANET*) [30,35] is a highly mobile network whose nodes are vehicles traveling along a road or a highway, and which communicate among themselves using short-range (e.g., 100-200 meters) wireless communication devices (such as Wi-Fi or UWB). Using *vehicle-to-vehicle* (*V2V*) communications [30] provides some interesting advantages. The most important benefit is probably that there is no need to deploy an expensive and wide-coverage support infrastructure. Moreover, as opposed to the case of mobile telephone networks, the users will not be charged for the use of such networks. Finally, there are applications that require a quick and direct exchange of data (without intermediate proxies or routers) between two vehicles within range of each other, for example in the area of safety.

Some works in the field also assume the existence of a fixed communication infrastructure that can be used to deliver information to vehicles or for vehicles to communicate with the fixed network, thus supporting *vehicle-to-infrastructure* (*V2I*) communications. This was especially true in early works for vehicular networks (e.g., in the VICS project[2]), but today many existing projects benefit from the use of V2V communications instead. Nevertheless, some roads may be equipped with some static relaying devices that provide Internet access to nearby vehicles by using a fixed network (e.g., see [36]), and so it can be interesting to use V2I communications when they are available.

Many opportunities appear to develop new applications for the vehicular field, especially for safety [31] and information purposes [10]. Thus, for example, vehicles can exchange data and receive information about different types of interesting events [10] (e.g., the traffic, emergency brakings, accidents, available resources such as parking spaces, etc.), or even be used to monitor the environment [16,42].

However, a number of difficulties also arise in this context, especially when V2V communications are used, due to the high mobility of the vehicles and the unreliability of the wireless communications in such a dynamic environment. So, new data management and query processing techniques are required for the drivers to effectively retrieve the information that is interesting for them, which could be provided by a variety of data sources.

3 Models for Accessing Data

In this section, we present different models that can be exploited by mobile devices in vehicular networks to access data.

3.1 Access to Remote Services

Today, wireless networks such as mobile telephone networks allow mobile users to connect to various remote databases, Internet sites, Web services, or services available on remote computers. These last years, some research works have also considered query processing using distributed data stored on remote devices, some of which focus on the use of *mobile Web services* [1] (i.e., Web services available on mobile devices).

An example of a remote service for drivers is the *Waze* application (`http://world.waze.com/`), which is a social mobile application which allows Waze users to publish and consume real-time maps and traffic information. Maps can be provided to mobile users and traffic information is retrieved using a mobile telephone network.

To use remote services in the evaluation of queries, one of the difficulties is to have a uniform representation of the available services. In a Web service context, the services that should be used to process a query can be retrieved by using a *UDDI* registry (*Universal Description, Discovery and Integration*, `http://uddi.xml.org/`), where each Web service is described with *WSDL* (*Web Services Description Language*, `http://www.w3.org/TR/wsdl`). However, in an heterogeneous context it is very difficult to have a standard representation of every available service. Another important problem is how to determine the services that are relevant for a given query.

[2] `http://www.vics.or.jp/english/vics/index.html`, Accessed May 9, 2011.

3.2 Data Dissemination – Push Model

Relying on a *push model* is a common approach for query processing in vehicular networks. Following a cooperative caching approach [34], each vehicle receives data from its neighbors and decides whether they are relevant enough to be stored in a local data cache or not [10]. Then, the data may be used locally by a query processor in charge of retrieving data relevant for the driver [44] (e.g., the driver may submit a query to retrieve the 5 nearest parking spaces). Thus, a query can only find an answer to a query *opportunistically*, that is, when a vehicle with the required information has passed nearby. The major difficulty for these solutions is how to disseminate data in the vehicular network so that vehicles receive the relevant information efficiently (timely and without unneeded overheads such as duplicate packets or irrelevant data).

Nevertheless, with a push model, not every data item will be communicated to every vehicle, as this would consume too much bandwidth and lead to high communication and processing efforts on the vehicles. On the contrary, only data about events that are potentially interesting for a large set of vehicles (e.g., an emergency braking or a traffic congestion) will be diffused among the vehicles that the system estimates as potentially interested. Information about other events will not be disseminated, and so it is impossible to share information among a small set of interested vehicles, for example to build vehicular social networks [39]. Moreover, the dissemination of a certain event is usually restricted to a spatio-temporal area where the event is estimated to be relevant. So, for example, a driver on a highway approaching a city would not be able to query which of the entries to the city she/he should take to avoid a traffic congestion.

3.3 Query Dissemination – Pull Model

Several works have considered a *pull model*, where a query is actually communicated to other vehicles in the vehicular network. This provides more flexibility in terms of the types of queries that can be considered, as opposed to the approaches based on a push model, since a query can in principle be diffused far away to retrieve remote data. This implies that vehicles should be able to understand, route, and process those queries.

The basic idea, inspired by traditional Peer-to-Peer (P2P) systems [2], consists of diffusing the queries to different data sources either directly or using multi-hop relaying techniques such as [7,28,15]. Then, each node can compute a partial query result based on its local data and then deliver it to the destination node. However, since no fixed data server or any kind of infrastructure is necessarily available in vehicular ad hoc networks, new techniques to access data are needed. Indeed, the mobility of nodes makes the management of an indexing structure, used in traditional P2P systems to decide how to route queries, impossible (as indicated in works such as [17]). Therefore, some mechanism must be used to route the queries to the vehicles that could store relevant information, usually based on spatio-temporal criteria (more appropriate in this context than using flooding or a random or biased walk [17]).

These works must also face the problem of routing the query results back to the query originator. This is a real challenge because the vehicle that issued the query can move in the meanwhile, and so routing the results based on simple geographic criteria may not be enough (it is difficult to know where the originator is currently located). Furthermore, since the vehicles keep moving, it is not even possible to ensure that there is at that moment a communication path to the originator node. A possibility to facilitate the routing process can be the use of mobile agent technology [41].

4 Multi-scale Mobile Query Processing

In the previous section, we have presented different models for accessing data in mobile environments. In the following, we show that these different models can be complementary. Indeed, in a mobile context, it can be very interesting not to focus on one particular access model but rather to consider *multi-scale query processing*, which implies exploiting the available data sources whatever the access mode (e.g., push or pull). The possible use of various data sources can first increase the number of potential results that can be retrieved and proposed to the user. It also increases the probability to provide users at least one useful result. Finally, combining several types of data sources allows to compute results that could not be computed otherwise or only with a lower quality.

4.1 Multi-Scale Queries: Definition and Examples

In [9], the author focuses on service-oriented environments and defines a *multi-scale query* as "a query over traditional datasets in conjunction with streaming data that may involve spatio-temporal aspects". In this paper, we generalize this definition and consider *multi-scale query processing* as any query processing that may need to access (possibly remote) data sources of different types to compute the result. To illustrate our purpose, we propose in the following several examples of mobile queries that may benefit from the multi-scale query evaluation process described in this paper:

- **Example 1:** *Retrieve all the parking spaces in a radius of 2 Km.* In this example, information about parking spaces can be extracted from a local data cache containing events received, through the use of short-range wireless networks, from neighboring vehicles leaving a parking space. Useful information about the location of parking lots can also be provided by different (Web) services accessed using mobile telephone networks (e.g., 3G). Considering a multi-scale processing scheme in this example, the query result would be composed by the union of both data sets (i.e., the locally computed one and the remote one/s). We can also note that both parts of the query (i.e., the local one and the remote one/s) can here be processed in parallel.
- **Example 2:** *Retrieve the list of petrol stations located in a radius of 10 Km around me where fuel prices are less than 1$ (and update the result every 5 minutes).* Different steps can be distinguished for this continuous query. One possible query processing scheme implies retrieving first the "petrol stations located in a radius of 10 Km". For this, a local database containing the locations of various points of interest (e.g., train stations, airports, restaurants, petrol stations, etc.) can be queried. Once the list of petrol stations within the specified radius has been retrieved, it is possible to compute the join between this data set and one containing the list of petrol stations where prices are lower than 1$ (obtained elsewhere).
- **Example 3:** *Retrieve the list of hotels with available rooms that I can reach in less than 30 minutes.* In this example, the list of hotels with available rooms can be provided by distant services. If the location is provided in terms of GPS coordinates, then the time to reach the target can be computed locally on the mobile device for each hotel retrieved (using a shortest path algorithm). If not, then the symbolic addresses of each hotel can be transformed into GPS coordinates using an additional Web service. Information provided by neighboring vehicles about the travel times and possible traffic congestions may also be useful to compute the answer to such queries.

The queries presented above illustrate the interest of obtaining information coming from different sources and how different alternative execution plans may be possible to process a given query.

4.2 Representation of Data Sources

To represent the properties of each data source, we advocate storing information about the different data sources accessible by the mobile device in an XML file. To decompose the query into several local queries to execute on the data sources, it is necessary to have precise information about the I/O parameters for each data source, the syntax of the local query language, the format of the results provided, etc. An example of this file is shown in Figure 1. The different data sources can be local or remote data sources. Examples of local data sources are:

- **Example 1:** An XML file stored locally on the mobile device, like the data source DS1 described in Figure 1. In this example, the local XML file contains items describing petrol stations. It can be updated with additional items broadcasted by other vehicles or petrol stations and received by the mobile device following a push model.
- **Example 2:** A local positioning service which provides to the user her/his location. As described in Figure 1, we consider in our example two different data sources providing the location of the mobile device: DS3 (a GPS interface) and DS4 (a Wi-Fi based positioning service). The XML element *connector* allows to define how the data source can be used. For example, the element *connector* "org.location.Gps" represents the name of a Java class which allows to retrieve GPS coordinates of mobile user.

In the following, we present some examples of remote data sources:

- **Example 1:** A Web service like DS2 in Figure 1, which provides information about gas stations according to a given symbolic address. In the figure, a description for the *EcoDrive.com* Web service, expressed using the Web Service Description Language (WSDL), is referenced. Such a WSDL description contains the different operations offered by the web service and the parameters to use to invoke them.
- **Example 2:** A Web service like DS5 in Figure 1, which allows to translate GPS coordinates into an address, and vice versa. This Web service can be used with a REST architecture [12]. Since no WSDL or equivalent description is available in this case, the different operations provided by the Web service and their parameters are listed in the XML file describing the data sources.

The previous examples show how the main features of the different data sources can be described in a flexible way by using an extensible language such as XML.

To simplify, we have considered here a local description of the data sources. However, provided that the services export their functionalities (for instance using WSDL), a dynamic discovery of the available services could be envisaged. Therefore, the query processor could query a remote directory to retrieve the set of useful services.

4.3 Semantic Equivalence

In addition to the information described in the previous section, it is also important to process queries to define the semantic equivalence between the data sources. In order to formulate queries, users indeed use a global conceptual schema providing the names of the different data items (e.g., *PetrolStation*) and their attributes (e.g., name, location, price, etc.). We thus follow a Global-as-View (GAV) approach for information integration [26]. However, contrary to traditional databases, the underlying data providers are highly heterogeneous in our case.

Some information is needed to decompose the user's query and to generate execution plans evaluated on remote Web services, local databases, etc. In our example, information about petrol stations can effectively be provided by remote Web services like DS2, which deals with gas stations, or retrieved from a local data cache aggregating the information about petrol stations received from other vehicles. To illustrate this, we describe in Figure 2 the representation of *PetrolStation* according to the different data sources presented in Figure 1. The data sources may provide several attributes, such as Name, Prices and Address. The attribute *Address* can be represented using two forms: *GPSCoordinates* or *SymbolicRepresentation* (such as the name, the street, the city etc.).

```
PetrolStation(Name, Price, Address(AddressRepresentation))
=
StationService(Nom, Prix, Adresse(GPSCoordinates))@DS1
UNION
GasStation(Name, Price, Address(SymbolicRepresentation))@DS2
```

Fig. 2 Example of the definition of equivalence between data sources

5 Optimization of Multi-scale Mobile Queries

Query processing can usually be decomposed in three different phases [23]. During the first one, the query is parsed and translated into an internal representation. The second phase aims at optimizing the query. Finally, the third phase implies executing the best plan generated in the previous phase. In the following, we detail how the list of the possible execution plans is generated and discuss the choice of the optimal query execution plan. We focus on the generation of different candidate queries, based on the selection of different data sources/services, as the optimization of the candidate queries themselves can be performed by using traditional query optimization techniques.

5.1 Generation of Possible Candidate Queries

To evaluate multi-scale queries, it is necessary to decompose the user's query into a set of sub-queries that have to be evaluated on different data sources (e.g., a local data cache, remote Web services, etc.). Therefore, different alternative query execution plans may be generated and considered for a single query. These execution plans contain both the data providers to use and the different evaluation steps which have to be performed according to the data sources selected to retrieve the query result. To illustrate the generation of possible execution plans to evaluate the query, let us consider the second example of query introduced in Section 4.1: *Retrieve the list of petrol stations located in a radius of 10 Km around me where fuel prices are less than 1$.* In a first step, we express the query using an SQL-like syntax (similar to what is proposed in [18]) as depicted in Figure 3.

```
SELECT PS.Name
FROM PetrolStation PS
WHERE PS.Price ≤ 1 AND inside(10, myloc, PS.Address)
```

Fig. 3 Example of a multi-scale mobile query formulated using an SQL-like language

The *inside* operator allows to process a constraint on the Euclidean distance between the client's physical location, called *myloc*, and the location of the petrol stations.

A first transformation consists in identifying the different data sources which are relevant to evaluate the query. For our example, we can first use two different solutions to obtain the user's location. Indeed, both the data sources DS3 and DS4 provide such an information: DS3 using GPS coordinates and DS4 using symbolic descriptions such as addresses. Moreover, to retrieve the information about petrol stations needed for our sample query, both the data sources DS1 and DS2 can be considered. The data source DS1 provides petrol station addresses using GPS coordinates. The data source DS2 allows to retrieve information about petrol stations

```xml
<?xml version="1.0" encoding="UTF-8"?>
<dataSources>
    <dataSource id="DS1" type="dataxml">
        <head>
            <title>Stations</title>
            <description>Gas station addresses and prices received by the mobile device using a push model.</description>
            <keywords>petrol station, gas station, prices, address</keywords>
        </head>
        <body>
            <XMLLocationFile>/Users/MyDocuments/Stations.xml</XMLLocationFile>
        </body>
    </dataSource>
    <dataSource id="DS2" type="web service">
        <head>
            <title>EcoDrive.com</title>
            <description>returns information about gas stations (prices and addresses) in a radius of n kilometers around a point represented
by a symbolic address</description>
        </head>
        <body>
            <wsdlDescription>http://www.EcoDrive.com/EcoDrive.wsdl</wsdlDescription>
        </body>
    </dataSource>
    <dataSource id="DS3" type="positioning service">
        <head>
            <title>GPS based positioning service</title>
            <description>returns mobile user position (GPS coordinates)</description>
        </head>
        <body>
            <connector>org.location.Gps</connector>
        </body>
    </dataSource>
    <dataSource id="DS4" type="positioning service">
        <head>
            <title>Wi-Fi based positioning service</title>
            <description>returns mobile user position (symbolic representation)</description>
        </head>
        <body>
            <connector>org.location.Wifi</connector>
        </body>
    </dataSource>
    <dataSource id="DS5" type="web service">
        <head>
            <title>maps.googleapis.com</title>
            <description>allows to translate gps coordinates to symbolic address or to translate symbolic address to GPS
coordinates</description>
        </head>
        <body>
            <operation name="LatLngToAddress">
                <url>http://maps.googleapis.com/maps/api/geocode/xml?latlng=latitudelongitude&sensor=true</url>
                <input>
                    <param1>latitude</param1>
                    <param2>longitude</param2>
                </input>
                <output>
                    <xmlfile/>
                </output>
            </operation>
            <operation name="AddressToLatLng">
                <url>http://maps.googleapis.com/maps/api/geocode/xml?address=address&sensor=true</url>
                <input>
                    <param1>address</param1>
                </input>
                <output>
                    <xmlfile/>
                </output>
            </operation>
        </body>
    </dataSource>
</dataSources>
```

Fig. 1 Example of the description of the data sources considered by a multi-scale mobile query processor

(for example the symbolic address of stations) within a certain distance around a point defined by its symbolic address. To compare two heterogeneous address representations, we need to use the data source DS5 which allows to translate a symbolic address to GPS coordinates, for example. To generate all possible candidate queries, we have to find all possible compositions of service calls. For each candidate query generated, the choice of the services used obviously have to respect implicit dependencies between services. Hence, when selecting the service which provides the user's location, the format of the location provided has to match with the expected format used to retrieve the nearby petrol stations. So, considering that the "distance" function used in the query is only dedicated to the comparison of locations based on GPS coordinates, the five candidate queries presented in Figure 4 are generated to satisfy the user's request.

The Candidate Query 1 presented in Figure 4 is based on the data source DS2. This data source represents a Web service which provides information about petrol stations based on the location of a mobile user and a distance parameter. This distance parameter defines the maximum distance between the location of the user (represented by a symbolic representation) and the location of the petrol station. The expression DS3.myloc allows to retrieve the location using the data source DS3 (i.e., using the GPS-based positioning service deployed on the mobile device). As the data source DS2 requires a symbolic representation of location user, the

```
Candidate Query 1
SELECT GS.Name
FROM DS2.GasStation(DS5. LatLngToAdress(DS3.myloc), 10) GS
WHERE GS.Price ≤ 1 ;

Candidate Query 2
SELECT GS.Name
FROM DS2.GasStation(DS4.myloc), 10) GS
WHERE GS.Price ≤ 1 ;

Candidate Query 3
SELECT StS.Nom
FROM DS1.StationService StS
WHERE StS.Prix ≤ 1
AND inside(10, DS3.myloc, StS.Adresse);

Candidate Query 4
SELECT StS.Nom
FROM DS1.StationService StS
WHERE StS.Prix ≤ 1
AND inside(10, DS5.AddressToLatLng(DS4.myloc), StS.Adresse);

Candidate Query 5
(SELECT GS.Name
FROM DS2.GasStation(DS5.LatLngToAdress(DS3.myloc), 10) GS
WHERE GS.Price ≤ 1 )
UNION
(SELECT StS.Nom
FROM DS1.StationService StS
WHERE StS.Prix ≤ 1
AND inside(10, DS3.myloc, StS.Adresse));
```

Fig. 4 Example of possible candidate queries

data source DS5 and the function *LatLngToAddress* are used to translate GPS coordinates to the symbolic location. The expression DS2.GasStation(DS5. LatLng-ToAdress(DS3.myloc), 10) allows to retrieve information about petrol stations located around the mobile user within a radius of 10 kilometers. The Candidate Query 2 is based on the same data source DS2 to retrieve information about gas stations. Nevertheless, in this query, the data source DS4 is used to retrieve the location of the mobile user. This data source provides the location of the user based on a symbolic representation. Thus, the web service described by the data source DS2 can directly used this location. Then, the Candidate Query 3 is based on the data source DS1. This data source relies on a XML file which stores information about petrol stations received using a push model. For each petrol station, the information are its name, its price, and its location as GPS coordinates. The Candidate Query 4 is based on the data source DS2 and the data source DS4 to retrieve the user location. So, the data source DS5 is used to translate the location of the mobile user into GPS coordinates. Finally, the Candidate Query 5 is based on the union of information retrieved using the data source DS1 and the data source DS2. Indeed, the Candidate Query 5 represents the union of the Candidate Query 1 and the Candidate Query 3. Other candidate queries based on the union of two subqueries (Query 1, Query 2, Query 3 and Query 4) can also be envisaged.

5.2 Query Optimization and Execution

By increasing the number of data sources that may be considered to compute the result, multi-scale queries can severely increase the processing cost. This is obviously an important issue, especially when dealing with devices with constrained resources. The role of the query optimizer is so crucial in this context. Moreover, the optimization objective used by most existing query processing systems is to minimize the *response time* (i.e., the amount of time the user will have to wait before obtaining her/his result), for example by reducing the number of disk I/Os or the network communications. Such optimization strategies are fully relevant in centralized environments. However, in environments where mobile devices are considered, even if the response time is still an important parameter to consider, it cannot be the only optimization objective. Thus, when processing multi-scale mobile queries, other parameters (such as the energy consumption of mobile devices like sensors [4]) have also a strong impact on the query evaluator.

To select the optimal candidate query, the first phase is to consider the right search space, that is to determine the set of services matching the user's expectations in terms of result quality. Obviously, the cost related to the evaluation of a multi-scale mobile query depends on the set of services used for processing that query. To choose this set of services, the user's expectations in terms of result quality have to be taken into account. For example, if the user accepts a "partial" result composed only of a small number of data items, only one data provider (ideally, the best one) needs to be contacted whereas all data sources providing the desired data items (e.g., petrol stations) should be queried if the user desires the most complete result (unless it is guaranteed that a complete result can be obtained by just querying some data sources).

The choice of the services to use may also depend on other quality parameters such as the accuracy of the information provided. For instance, when dealing with location queries, some of the services may not provide a sufficient accuracy to process the constraint defined by the user (e.g., distance < 100 m).

Both to select the services according to quality expectations and to compute the costs related to the evaluation of a multi-scale mobile query, a set of properties is needed to characterize each data source. More precisely, the following attributes are useful for the optimization process:

– *Accuracy*: level of accuracy (e.g., low, medium or high) of the location information possibly provided by the data source;

- *Average availability*: percentage indicating the reliability of the data source (e.g., the data source is available 90% of the time);
- *Cost*: financial cost related to the use of the data source. The value of this attribute is set to 0 if the use of the data source is free;
- *Data production rate*: average quantity of information delivered by the data source in a query result;
- *Response time*: average time needed by the remote data source to process a query;
- *Update frequency*: average refreshment level of the information provided by the data source (e.g., average age of the fuel prices provided by the data source).

Some of these properties may be exported by the data source, whereas others need to be monitored by an external service. Besides, these properties are not only used to select the services to consider for the evaluation of a query but also to compute the costs associated to each candidate query in order to select and execute the optimal one. To evaluate the cost of a query, we consider three different dimensions:

- **Time** - Obviously, the *time needed to compute the query result* is still an important criterion to consider in mobile environments even if it is no more the only optimization objective.
- **Money** - The *financial cost* associated to the evaluation of a particular query has been considered by several query optimizers in the field of distributed databases [47]. In the context considered in this paper, the money spent to process a query may be related to the use of some types of networks (e.g., mobile telephone networks) or some premium services.
- **Energy** - The *energy cost* (i.e., the energy spent by a mobile device to process a query) is particularly important when dealing with mobile devices, whose autonomy is limited [40].

The different costs mentioned above are used by the query optimizer to choose, among the list of possible execution plans, the one providing the best compromise. Therefore, the user has here to express preferences about how the different costs considered have to be balanced (e.g., the energy consumption and the financial costs should be considered as more important than the time needed to compute the result).

We use the following cost functions to determine the cost of a query on each dimension:

$$C_{Time}(Q) = C^{Time}_{QueryDelivery}(Q) + C^{Time}_{Processing}(Q) + C^{Time}_{ResultDelivery}(Q)$$

$$C_{Money}(Q) = C^{Money}_{QueryDelivery}(Q) + C^{Money}_{Processing}(Q) + C^{Money}_{ResultDelivery}(Q)$$

$$C_{Energy}(Q) = K \times n$$

where n is the number of bytes to transfer during the evaluation of the query Q and K is a constant[3].

Finally, the global cost of each query is obtained by computing a weighted sum for the different costs:

$$C(Q) = \sum_{i=1}^{3} \omega_i \times C_i$$

where ω_i is the weighting factor assigned to cost C_i, used to balance the relative importance of the different costs. The range of ω_i is [0,1] and $\Sigma\omega_i = 1$. The goal of the query optimization is to minimize $C(Q)$, based on statistics collected.

6 Prototype Implemented

To validate our approach, we have developed a prototype of a multi-scale mobile query processor for vehicular networks. We developed the prototype of the query processor in the context of our *VESPA* (*Vehicular Event Sharing with a mobile P2P Architecture*) system [10], which is a system designed for vehicles to share information (e.g., available parking spaces, accidents, emergency brakings, obstacles in the road, real-time traffic information, information relative to the coordination of vehicles in emergency situations, etc.) in inter-vehicle ad-hoc networks. A prototype of VESPA has be developed using *Microsoft .NET*. We have evaluated the prototype in a small-scale using real vehicles and in a large-scale through simulations (for more details, see `http://www.univ-valenciennes.fr/ROI/SID/tdelot/vespa/`).

In the VESPA system, an event is relevant for the vehicle if the vehicle will probably meet the event. An event is relevant for the driver if it is relevant for her/his vehicle and also if the driver is interested in that type of event. Therefore, continuous query processing techniques are needed. Moreover, to provide users the maximum number of relevant data items, we have extended our query processor with multi-scale access features, as described earlier in this paper (see the examples of queries provided in Section 4.1). For the implementation, we studied the possibility to use the *Microsoft*

[3] To process the *energy cost*, we rely on the data provided by [11] stating that the energy consumption can be approximated by $44 \times 10^{-6} \times n$ (Joules).

LINQ API . In the following, we first present the LINQ API and then some basic aspects of the prototype implemented.

6.1 The Microsoft LINQ API

Language Integrated Query (*LINQ*) introduces a set of features in the *Microsoft .NET Framework 3.5* by providing query capabilities to the syntax of the .NET languages, especially C#. LINQ indeed introduces standard patterns for querying and updating data. Moreover, it can be used to access several types of data stores. By default, the .NET Framework 3.5 includes several LINQ provider assemblies that enable the use of LINQ with four types of data sources: *.NET Framework collections*, *SQL Server databases*, *XML documents*, and *ADO.NET datasets*.

Thus, one of the benefits of LINQ is the possibility to query different types of data sources (a data structure, a Web service, a file system, or a database) which may be local or remote. For this purpose, it provides different interfaces (e.g., *IEnumerable*, *IQueryable*) that ensure that the LINQ queries can be either evaluated on the target data sources or translated to other types of queries supported by the corresponding data source (e.g., SQL queries).

6.2 Prototype of a Multi-Scale Query Processor

To design a prototype of a multi-scale mobile query processor using Microsoft .NET, one challenge was to perform a continuous processing, which is basically not supported by LINQ. Therefore, we decided to combine LINQ with the *Windows Presentation Foundation* (*WPF*). Indeed, thanks to the data binding capabilities provided by WPF, it is possible to implement arrival-based query processing mechanisms. Thus, when new data items arrive in the local data cache, each continuous query under evaluation is immediately notified by this event. If the item received is relevant, then it is included in the result set of the continuous query; otherwise, it is simply ignored.

However, some specific queries may require data which are not currently shared by the vehicles; in other words, they may require data which will be never stored in the local cache. Therefore, the query processor needs the capability to obtain extra information from remote Web services. Indeed, for each kind of information needed, there probably exists a Web service which could provide the relevant data. LINQ facilitates the tasks to request data from such Web services, as it allows accessing any kind of data source.

We have developed two types of external LINQ providers. The first type is able to translate a LINQ query into an equivalent HTTP GET request (with the appropriate parameters) to the Web server. The second one is able to transform a LINQ query into a corresponding method call to a specific Web service using the SOAP protocol.

6.3 Sample Scenario: Measuring the Costs of Query Execution

In this section, we illustrate the importance of our query optimizer in a muti-scale environment based on a real example using heterogeneous data sources (local or remote). In our example, we consider a driver who wants to find the cheapest petrol station in a radius of 10 Km around her/him. She/he therefore uses our prototype of multi-scale mobile query processor deployed on a smartphone equipped with a 3G connection. To process the query, information about petrol stations can be retrieved from a large set of Web services. To simplify, we only consider the following data sources in our example:

– the *Carbu.fr* Web service, called DS1, which provides information such as prices or available services (shop, carwash etc.) about gas stations around the location of the mobile user;
– the *Prix-carburants.gouv.fr* Web service, called DS2, which provides information (prices and address) about the gas stations located in the same city than the mobile user;
– the local data cache, called DS3, containing the location of the different petrol stations. This information can be for example extracted from the navigation device using the petrol stations "points of interest".
– the data source DS4 providing the location of the mobile user.

To compute the result of the user's query, the query processor has to retrieve the list of petrol stations with the fuel prices proposed using a Web service (DS1 or DS2). It has also to determine the set of petrol stations in a radius of 10 Km around the user using DS3 and DS4. Finally, by computing the join between both result sets on the petrol station *name* attribute and selecting the cheapest one, the query result is obtained. As for the example described in Section 5.1, several candidate queries can be generated to process the user's query. For instance, a first candidate query (CQ1) can be defined using data source DS1 and a second one (CQ2) using the data provided by DS2. Other solutions are possible

	Latency stationary (seconds)	Latency mobile (seconds)	Data production rate (KB)
DS1	8.73	9.37	277
DS2	4.24	5.31	85

Table 1 Statistics provided by the monitoring service for DS1 ("Carbu.fr") and DS2 ("Prix-carburants.gouv.fr")

(e.g., use both the information provided by DS1 and DS2 to compute the query result), but we will focus on these two queries in the following. Among the different execution plans, the query optimizer has to determine the best execution plan according to the preferences (i.e., the weighting factors) defined by the user.

In this example, the possible difference between these plans only relies on the choice of the remote data source used to process the query (i.e., both execution plans share the same local work). In the following, we illustrate on real data how our multi-scale query processor can evaluate the cost of each solution in such a case to select the best one and process it. For simplicity reasons, we do not consider any financial cost in this example.

To determine the cost of each possible candidate query, the query optimizer uses the information provided by a monitoring service, which role is to provide the necessary statistics. In this example, we notably use the *Latency* of the data sources and their *Data production rate*. The *Data production rate* corresponds to the quantity of information delivered by the data sources. This information is obtained by processing a set of queries on the different data sources. The useful statistics in our example are presented in Table 1. In our prototype, these results have been obtained by performing one hundred of measurements. To obtain these statistics, we assumed that wireless transfers were performed with a 3G connection. We also evaluated the impact of mobility on the *Latency* by considering both stationary users and users driving their car with an average speed of 50 km/h. Thus, our experimentations have highlighted an increase of the *Latency* when the user is moving.

Thanks to the statistics and metadata about services, a cost can be computed for each dimension (i.e., Time, Energy and Money). As indicated in Section 5, a weighting factor can also be associated to each cost. To identify the optimal candidate query according to the user's expectations, a global cost is thus computed for each candidate query. To illustrate the choice of optimal candidate query, we chose to use the same weighting factor for each cost ($\omega_{Time} = \omega_{Energy} = 0,5$). In Table 2, we summarize the costs computed for both considered candidate queries. The cost in terms of time is computed using the statistics described in the Table 1 and the time cost of the join operation (i.e. 3.68 seconds using DS1 and 1.65 seconds using DS2). The cost in terms of energy consumption is computed using the average number of bytes transferred, as described in Section 5.

	$Cost_{Time}$	$Cost_{Energy}$	Global Cost
CQ1	13.05	12.48	12.76
CQ2	6.96	3.81	5.38

Table 2 Costs computed for CQ1 and CQ2

Thanks to the information provided by the monitoring service, the query optimizer is able to select the execution plan corresponding to CQ2 since it strongly minimizes the global cost. Nevertheless, it is also very important to integrate the expected result's quality when choosing the best candidate query. Even if both data sources considered in this example provide a consequent amount of data items (i.e., petrol stations), the strategy presented here can lead to selecting the cheapest execution plan, even if the quality of the result provided is very low. To avoid this, the user's expectations in terms of quality have to be integrated, for example by using the properties describing the data sources introduced in section 5.2.

7 Related Work

Several works have considered query processing issues in *mobile P2P* (*Peer-to-Peer*) environments or mobile wireless sensor networks (e.g., [14,17]). Even though these works are related to this paper, the high mobility of vehicles and the special features of vehicular networks introduce additional challenges. Therefore, we focus in this section mostly on works in the field of vehicular networks or in closely related contexts. We consider the approaches in two different categories depending on whether they advocate a push model (see Section 3.2) or a pull model (see Section 3.3)[4], with an emphasis on the approaches proposing a pull mode because of the particular challenges that they must tackle. It is not our goal to provide an exhaustive list of related works or analyze them in depth, but just to offer a general overview about existing approaches.

Before describing the related work, it is important to stress that existing approaches usually focus on a spe-

[4] Only a few works consider the access to remote services in the context of vehicular networks (see Section 3.1 for an example), and so they are not described here.

cific scenario. Moreover, we are not aware of any work in the field of vehicular networks that tries to perform a multi-scale query processing accessing different and heterogeneous data sources.

7.1 Works Based on a Push Model

The *Mobi-Dik* project [44], motivated by the interest of sharing information about available parking spaces, proposes an *opportunistic exchange* mechanism (vehicles communicate with one another when they are close) inspired by the field of epidemiology. A vehicle with a certain piece of information acts as a disease carrier, and "contaminates" the nearby vehicles along its route. Once contaminated, these vehicles proceed to contaminate others. This dissemination principle is accompanied by mechanisms that monitor the relevance of the information (based on temporal and spatial criteria) in order to decide if it should be stored in the cache and/or broadcasted later on.

In [33], the focus is on the dissemination of information about road hazards. As in Mobi-Dik, an opportunistic exchange is proposed and three different communication strategies to decide when a peer should communicate information to another peer in its neighborhood are described and compared: the classical *flooding* strategy, the *epidemic* strategy (inform only a certain number of peers), and the *proximity* strategy (inform only the peers within a certain distance of the location of the event).

In [10], the VESPA (Vehicular Event Sharing with a mobile Peer-to-peer Architecture) project is presented, which is based on the concept of *Encounter Probability* for vehicles to share information using vehicle-to-vehicle communications. The goal is to facilitate the dissemination of information in the vehicular network and also to determine when certain information should be shown to the driver, taking into account the relevance of the data. This is a general approach that does not focus on a particular type of event, as it supports in a uniform framework different types of events including mobile events (e.g., an event indicating an ambulance asking other vehicles to yield the right of way).

The above are just some examples of works based on a push model, but there are others such as [27,29].

7.2 Works Based on a Pull Model

PeopleNet [32] is an infrastructure-based proposal for information exchange in a mobile environment. However, it relies on the existence of a fixed network infrastructure to send a query to an area that may contain relevant information. Even though, once the query has arrived in the target area, it uses epidemic query dissemination through short-range communications within the area (to save economic costs), the answer to the query is communicated again to the originator using the fixed network (e.g., by sending an SMS or email). Thus, problems related to query routing and result routing do not appear in this context. Moreover, this proposal does not focus on vehicular networks.

FleaNet [25] is a virtual market organized over a vehicular network. It proposes a *mobility assisted query dissemination* where the node that submitted the query periodically advertises it only to its one-hop neighbors, which will see if they can provide some answers from information stored on their local caches. With this approach the query spreads only due to the motion of the vehicle that submitted the query. This avoids overloading the network with many messages. It also solves the problem of routing back the results to the query originator, since it is only at one-hop distance. However, it is not general enough to process some kinds of queries, since the vehicle that submits the query must move near the vehicles which store the information needed.

In [48], the authors introduce *Roadcast*, a content sharing scheme for VANETs. As in FleaNet, a vehicle can only query other vehicles that it encounters on the way. So, the problems of query and result routing do not arise either. In this case, keyword-based queries are submitted by the users and the scheme proposed tries to return the most popular content relevant to the query, as this content is likely to be useful for more vehicles in the future. Thus, not necessarily the most relevant data are returned for a query, as the popularity of the data is also taken into account.

In [45], a combination of pull and push is considered for *in-network query processing in mobile P2P databases*. When two mobile nodes encounter each other, they first exchange queries and results (pull phase). Then, they broadcast other popular data items that may help the other peer to improve its capabilities as source of relevant data in the future (push phase). Once again, multi-hop query/result routing is not considered.

Finally, in [46] the problem of searching documents in a vehicular network is considered. In this approach, the authors adapt the concept of *Distributed Hash Table* (*DHT*) [2] to a mobile environment and propose a *Hybrid Retrieval* (*HR*) approach which, based on the expected costs, adapts itself to choose between a flooding scheme and a DHT scheme for indexing and searching. The geographic space is divided into regions, such that each region must keep certain data. So, when a vehicle leaves a region it must transfer the documents belonging to the region to some other vehicle within the region. Thus, the purpose and approach of this work is quite different from those traditionally found in the context of vehicular networks. Besides, the problem of routing the results to the query originator is not discussed.

8 Conclusion and Perspectives

Vehicular networks open up new opportunities to develop different data services that provide drivers with interesting information. However, they also bring some challenges. In particular, new data management and query processing techniques are required. One interesting idea, which we call *multi-scale* query processing, implies to be able to combine different access modes to retrieve data (e.g., push and pull) from different data sources (e.g., data stored locally, data available in nearby vehicles, data provided by Web services, etc.). The data retrieved from the different data sources must then be combined to compose the final answer to the query. However, key research questions arise, for example from the point of view of the selection of the relevant data sources, the consideration of different alternative execution plans, and the correlation of the data retrieved. Moreover, some access modes are not so easy to carry out in the context of vehicular ad hoc networks, as it is the case for pull approaches, where query and result routing problems appear.

In this paper, we have described multi-scale mobile query processing and optimization techniques. Several examples have been used to illustrate both the benefits and the difficulties involved. Moreover, we have also presented a prototype of a first query evaluator developed using the Microsoft LINQ API. As far as we know, no other work has considered this problem of multi-scale query processing in vehicular networks.

As future work, we believe that both the query optimization phase and the query execution phase have to take into consideration the execution context in order to effectively evaluate queries. Thus, the query processor should have the capability to face the frequent changes inherent to the mobile and dynamic environment considered in this paper. More precisely, it should adapt itself to the current situation in order to ensure a *Quality of Service* (*QoS*) that fulfills the user preferences, the constraints of the devices (e.g., energy available, processing capabilities, etc.), and the network characteristics (e.g., the availability of a mobile telephone network, the available bandwidth, etc.). Currently, we are studying the problem of query/result routing in vehicular ad hoc networks.

Acknowledgements This research work has been partially supported by the French research agency (ANR) in the scope of the OPTIMACS project, and the CICYT projects TIN2007-68091-C02-02 and TIN2010-21387-C02-02.

References

1. Mustafa Adacal and Ayse Bener. Mobile web services: A new agent-based framework. *IEEE Internet Computing*, 10(3):58–65, 2006.
2. Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys*, 36(4):335–371, 2004.
3. Arvind Arasu, Shivnath Babu, and Jennifer Widom. The CQL continuous query language: Semantic foundations and query execution. *VLDB Journal*, 15:121–142, 2003.
4. Fuad Bajaber and Irfan Awan. Energy efficient clustering protocol to enhance lifetime of wireless sensor network. *Journal of Ambient Intelligence and Humanized Computing*, 1(4):239–248, 2010. 10.1007/s12652-010-0019-x.
5. Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. A survey on context-aware systems. *Int. Journal of Ad Hoc and Ubiquitous Computing*, 2(4):263–277, June 2006.
6. Daniel Barbará. Mobile computing and databases – a survey. *IEEE Transactions on Knowledge and Data Engineering*, 11(1):108–117, 1999.
7. Sanjit Biswas and Robert Morris. ExOR: opportunistic multi-hop routing for wireless networks. *ACM SIGCOMM Computer Communication Review*, 35(4):133–144, 2005.
8. Angela Bonifati, Panos K. Chrysanthis, Aris M. Ouksel, and Kai-Uwe Sattler. Distributed databases and peer-to-peer databases: past and present. *SIGMOD Record*, 37(1):5–11, 2008.
9. Victor Cuevas-Vicenttin. Towards multi-scale query processing. In *IEEE 24th Int. Conf. on Data Engineering (ICDE'08) Workshops*, pages 137–144. IEEE Computer Society, 2008.
10. Thierry Delot, Nicolas Cenerario, and Sergio Ilarri. Vehicular event sharing with a mobile peer-to-peer architecture. *Transportation Research Part C: Emerging Technologies*, 18(4):584–598, 2010.
11. Laura Marie Feeney and Martin Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *Int. Conf. on Computer Communications (INFOCOM'01)*. IEEE Computer Society, 2001.
12. Roy T. Fielding and Richard N. Taylor. Principled design of the modern Web architecture. *ACM Transactions on Internet Technology*, 2(2):115–150, May 2002.
13. Hocine Grine, Thierry Delot, and Sylvain Lecomte. Adaptive query processing in mobile environment. In *Proceedings of the ACM Third Int. Workshop on Middleware for Pervasive and Ad-Hoc Computing*, 2005.
14. Yingjie He and Alan Tully. Query processing for mobile wireless sensor networks: State-of-the-art and research challenges. In *Third Int. Symposium on Wireless Pervasive Computing (ISWPC'08)*, pages 518–523. IEEE Computer Society, 2008.
15. Masahiro Hiyama, Makoto Ikeda, Leonard Barolli, and Makoto Takizawa. Performance analysis of multi-hop ad-hoc network using multi-flow traffic for indoor scenarios. *Journal of Ambient Intelligence and Humanized Computing*, 1(4):283–293, 2010. 10.1007/s12652-010-0021-3.

16. Shu-Chiung Hu, You-Chiun Wang, Chiuan-Yu Huang, Yu-Chee Tseng, Lun-Chia Kuo, and Chao-Yu Chen. Vehicular sensing system for CO2 monitoring applications. In *IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS'09)*, pages 168–171, 2009.
17. Huilong Huang, John H. Hartman, and Terril N. Hurst. Efficient and robust query processing for mobile wireless sensor networks. *Int. Journal of Sensor Networks*, 2(1/2):99–107, 2007.
18. Sergio Ilarri, Eduardo Mena, and Arantza Illarramendi. Location-dependent queries in mobile contexts: Distributed processing using mobile agents. *IEEE Transactions on Mobile Computing*, 5(8):1029–1043, August 2006.
19. Sergio Ilarri, Eduardo Mena, and Arantza Illarramendi. Location-dependent query processing: Where we are and where we are heading. *ACM Computing Surveys*, 42(3):1–73, March 2010.
20. Tomasz Imielinski. *Mobile Computing*. Kluwer, Boston, MA, USA, 1996.
21. Tomasz Imielinski and Badri Nath. Data management for mobile computing. *SIGMOD Record*, 22(1), 1993.
22. Tomasz Imielinski and Badri Nath. Wireless graffiti: data, data everywhere. In *Int. Conf. on Very Large Data Bases (VLDB'02)*, pages 9–19. VLDB Endowment, 2002.
23. Donald Kossmann. The state of the art in distributed query processing. *ACM Computing Surveys*, 32(4):422–469, 2000.
24. Manolis Koubarakis, Timos Sellis, Andrew Frank, Stphane Grumbach, Ralf Hartmut Güting, Christian Jensen, Nikos Lorentzos, Yannis Manolopoulos, Enrico Nardelli, Barbara Pernici, Hans-Jörg Schek, Michel Scholl, Babis Theodoulidis, and Nectaria Tryfona, editors. *Spatio-Temporal Databases: The CHOROCHRONOS Approach*. Springer, 2003.
25. Uichin Lee, Jiyeon Lee, Joon-Sang Park, and Mario Gerla. FleaNet: A virtual market place on vehicular networks. *IEEE Transactions on Vehicular Technology*, 59(1):344–355, 2010.
26. Maurizio Lenzerini. Data integration: a theoretical perspective. In *21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'02)*, pages 233–246. ACM, 2002.
27. Ilias Leontiadis, Paolo Costa, and Cecilia Mascolo. Persistent content-based information dissemination in hybrid vehicular networks. In *Seventh IEEE Int. Conf. on Pervasive Computing and Communications (PerCom'09)*. IEEE Computer Society, 2009.
28. Christian Lochert, Hannes Hartenstein, Jing Tian, Holger Füßler, Dagmar Hermann, and Martin Mauve. A routing strategy for vehicular ad hoc networks in city environments. In *Intelligent Vehicles Symposium (IV'03)*, pages 156–161. IEEE Computer Society, 2003.
29. Christian Lochert, Björn Scheuermann, Murat Caliskan, and Martin Mauve. The feasibility of information dissemination in vehicular ad-hoc networks. In *Forth Conf. on Wireless on Demand Network Systems and Services (WONS07)*, pages 92–99. IEEE Computer Society, 2007.
30. Jun Luo and Jean-Pierre Hubaux. A survey of research in inter-vehicle communications. In *Embedded Security in Cars – Securing Current and Future Automotive IT Applications*, pages 111–122. Springer, 2005.
31. Christophe J. Merlin and Wendi B. Heinzelman. A study of safety applications in vehicular networks. In *IEEE Int. Conf. on Mobile Adhoc and Sensor Systems (MAHSS'09)*. IEEE Computer Society, 2009.
32. Mehul Motani, Vikram Srinivasan, and Pavan S. Nuggehalli. PeopleNet: engineering a wireless virtual social network. In *11th Annual Int. Conf. on Mobile Computing and Networking (MobiCom'05)*, pages 243–257. ACM, 2005.
33. Silvia Nittel, Matt Duckham, and Lars Kulik. Information dissemination in mobile ad-hoc geosensor networks. In *Int. Conf. on Geographic Information Science (GIScience'04)*, volume 3234 of *Lecture Notes in Computer Science*, pages 206–222. Springer, 2004.
34. Thabo Nkwe, Mieso Denko, and Jason Ernst. Data ubiquity in autonomic wireless mesh networks. *Journal of Ambient Intelligence and Humanized Computing*, 1(1):3–13, 2010. 10.1007/s12652-009-0001-7.
35. Stephan Olariu and Michele C. Weigle, editors. *Vehicular Networks: From Theory to Practice*. Chapman & Hall/CRC, 2009.
36. Jörg Ott and Dirk Kutscher. Drive thru internet: IEEE 802.11 for "automobile" users. In *23rd Int. Conf. on Computer Communications (INFOCOM'04)*, pages 362–373. IEEE Computer Society, March 2004.
37. Evaggelia Pitoura and George Samaras. *Data Management for Mobile Computing*. Kluwer Academic Publishers, 1998.
38. Ayse Y. Seydim, Margaret H. Dunham, and Vijay Kumar. Location dependent query processing. In *Second ACM Int. Workshop on Data Engineering for Wireless and Mobile Access (MobiDE'01), collocated with ACM SIGMOD*, pages 47–53. ACM, 2001.
39. Stephen Smaldone, Lu Han, Pravin Shankar, and Liviu Iftode. RoadSpeak: enabling voice chat on roadways using vehicular social networks. In *First Workshop on Social Network Systems (SocialNets'08)*. ACM, 2008.
40. Marie Thilliez and Thierry Delot. Evaluating location dependent queries using ISLANDS. In *Advanced Distributed Systems*, volume 3061 of *Lecture Notes in Computer Science*, pages 125–136. Springer, 2004.
41. Oscar Urra, Sergio Ilarri, Thierry Delot, and Eduardo Mena. Mobile agents in vehicular networks: Taking a first ride. In *Eight Int. Conf. on Practical Applications of Agents and Multi-Agent Systems (PAAMS 2010)*, volume 70 of *Advances in Intelligent and Soft Computing*, pages 118–124. Springer, April 2010.
42. Oscar Urra, Sergio Ilarri, Eduardo Mena, and Thierry Delot. Using hitchhiker mobile agents for environment monitoring. In *Seventh Int. Conf. on Practical Applications of Agents and Multi-Agent Systems (PAAMS'09)*, volume 55 of *Advances in Intelligent and Soft Computing*, pages 557–566. Springer, March 2009.
43. Genoveva Vargas-Solar, Noha Ibrahim, Christine Collet, Michel Adiba, Jean-Marc Petit, and Thierry Delot. *Pervasive Computing and Communications Design and Deployment: Technologies, Trends, and Applications*, chapter Querying Issues in Pervasive Environments. IGI Global, 2010.
44. Bo Xu, Aris Ouksel, and Ouri Wolfson. Opportunistic resource exchange in inter-vehicle ad-hoc networks. In *Fifth Int. Conf. on Mobile Data Management (MDM'04)*, pages 4–12. IEEE Computer Society, 2004.
45. Bo Xu, Fatemeh Vafaee, and Ouri Wolfson. In-network query processing in mobile P2P databases. In *ACM Int. Conf. on Advances in Geographic Information Systems (GIS'09)*, pages 207–216. ACM, 2009.
46. Quanqing Xu, Heng Tao Shen, Zaiben Chen, Bin Cui, Xiaofang Zhou, and Yafei Dai. Hybrid retrieval mechanisms in vehicle-based P2P networks. In *Int. Conf. on Computational Science (ICCS'09)*, volume 5544 of *Lecture*

Notes in Computer Science, pages 303–314. Springer, 2009.

47. Haiwei Ye, Brigitte Kerherv, Gregor v. Bochmann, and Vincent Oria. Pushing quality of service information and requirements into global query optimization. In *Int. Database Engineering and Applications Symp. (IDEAS'03)*, pages 170–179. IEEE Computer Society, 2003.
48. Yang Zhang, Jing Zhao, and Guohong Cao. Roadcast: A popularity aware content sharing scheme in VANETs. In *29th IEEE Int. Conf. on Distributed Computing Systems (ICDCS'09)*, pages 223–230. IEEE Computer Society, 2009.