Service-Orientation in Mobile Computing - An Overview

Mohamed Hamdy
Faculty of Computer and Information Sciences, Ain Shams University.
Cairo, Egypt - 11566.
mohx@email.com

Birgitta König-Ries
Friedrich-Schiller-Universität Jena
Ernst-Abbe-Platz 2
07743 Jena, Germany
koenig@informatik.uni-jena.de

Abstract

In this paper, we argue why service-orientation is the appropriate computing paradigm to use in mobile and wireless environments. We explain how the limitations of mobile devices can be overcome by functionality sharing via service-orientation. We identify some key areas that need to be addressed in order for this vision to become a reality. We survey existing approaches in these areas and discuss what is still missing.

Keywords:
Mobile computing, service orientation, query processing

1- Introduction

Over the last few years, wireless communications and mobile computing have become increasingly more popular and widespread.

In mobile environments which are composed of cellular technologies or ad-hoc technologies or mixed approaches, there is a collection of mobile heterogeneous hosts, which are enabled to communicate using "wireless links". These wireless links may change according to the natures of mobile networks, moreover, nodes in the ad-hoc network have to communicate without any centralized or help [1]. Each mobile node offers limited functionality only. However, as a whole, the devices can handle more complex tasks. Thus, mechanisms that allow the sharing of functionality among different – mobile or stationary – devices are needed.

Consider as an example the following situation: Anna, a computer scientist, travels to a conference to give a talk. Since she likes to travel light, she only brings her smart phone along. When she arrives at her destination airport, she accesses local information sources (either official ones or information provided by fellow travelers) to obtain an electronic map of the city and information about taxi rates to her hotel. The next day, at the conference, she uses an application provided by the conference organizers that lets her know, which other participants have interests similar to her own and alerts her when any of those are in her vicinity. For her talk, Anna "tells" the system in the meeting room that she wants to use a display medium and have the lights adjusted to optimize the viewing of her slides. Anna could achieve all this by manually looking for appropriate information sources and addressing them – however, this would be cumbersome and unattractive.

The service-oriented computing paradigm offers a means to achieve the functionality described above without the need for manual intervention: Devices describe the functionality they can offer to the network in terms of services. Examples for such services could be the provision of a display medium, the ability to use CPU time, or access to information stored on a device. On the other hand, devices can describe their needs in terms of service requests. A device could for instance request a service providing information regarding train connections from Tokyo to Kyoto or request a service converting a document from postscript to PDF format.

Figure 1 depicts a generic architecture for service-oriented applications, also known as the "service triangle". In a first step, someone who is willing to share functionality or information, i.e., a service provider compiles a service description. This description contains information about what is offered and how it can be accessed. This description is made available in a (centralized or distributed) service repository. Whenever someone (a service requestor) needs information or

functionality that is not available on her own device, she will formulate a service request and send it to the repository.

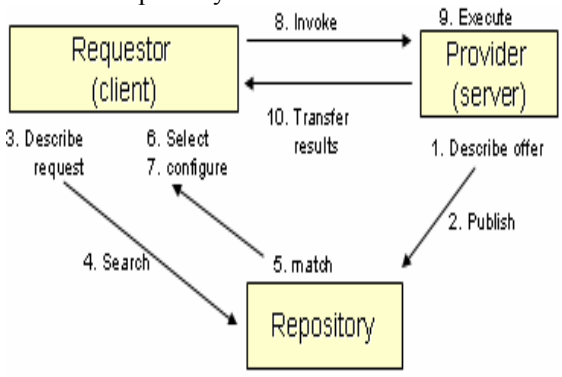

Figure 1: Service triangle

It is compared to the available offers and matching results are returned to the requestor. The requestor then chooses the most appropriate of the proposed offers, configures it according to her needs and invokes the service at the service provider's site.

Service-orientation will only be beneficial in our scenario, if it allows for full automation. In particular, it must be possible to automatically match offers and requests and to automatically configure and invoke services. Furthermore, it must be feasible to execute these tasks on mobile devices.

The first prerequisite for automation is descriptions of offers and requests that contain enough information to automatically decide whether a given offer is fulfilled by a given request. Often, it will not be possible to fulfill a request with a single service. Rather, different services need to be combined (or composed) in order to achieve the desired functionality.

A number of factors play a role in finding the optimal execution with composed services. Maybe the same functionality is offered by two devices. Which one should be used? The answer may depend on the current position of the devices in the network, their battery levels and so on. Maybe functionality from two devices needs to be combined. Which service should be accessed first? Where the results should be combined? These questions are somewhat similar to those addressed in query optimization as requests can be viewed as queries; however, classical query optimization methods can not be directly applied to the more restricted mobile environments [2].

In the remainder of this paper, we take a closer look at these key areas: a) service descriptions (Section 2), b) service discovery and composition (Section 3), and c) service execution (Section 4). For each of these areas, we describe existing approaches and discuss what remains to be solved to make them usable in a mobile environment. We conclude the paper with a summary in Section 5.

2- Service Description

Accessing data and functionality is obtained by posing queries (service requests) to be matched with service offers. The more precise queries and offers are described, the better matches can be achieved. A good service description language is therefore the most important basis for service-oriented architectures [3]. In this section we take a look at ways of describing offers and requests and investigate how well existing approaches are suited for the mobile environment.

2.1. Describing Service Offers

2.1.1 Existing Approaches

WSDL (Web Services Description Language) [4] is an XML-based language for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. WSDL offers a valuable basis for service orientation, since it allows describing services very well on a technical level. Also, it enjoys wide support by major industry players. However, WSDL fails to describe the functionality of the service in a machine-understandable way. This functionality can at best – and maybe wrongly – be guessed from the description of the message flow. Accordingly, query matchers are primitive, automatic service selection is not possible.

OWL-S (Web Ontology Language-Services) [5] has many improvements over WSDL such as well defined semantics, or separation of semantics from concrete descriptions. The OWL-S ServiceProfile contains information about non-functional properties like contact information, QoS parameters and so on, and a functional description via inputs, outputs, preconditions, and results (IOPRs) consisting mainly of references to the ServiceModel. The goal of this model is to describe the procedure that is necessary to interact with the service from a client's point of view and to provide information about the inner workings of the service. The main drawbacks of OWL-S are the scattering of the description of the functional semantics across several parts of the service description and a lack of sophisticated service matchers.

WSMO (Web Service Modeling Ontology) [6], is a conceptual model for the description of semantic web services together with a family of underlying formal languages (WSML). It offers strict decoupling (resources are described independent of each other), ontological role separation (i.e., separate constructs for all offers and requests) and uses web service specification as the service's processing scheme. Moreover, by applying different types of mediators WSMO can overcome heterogeneity at different levels.

DSD (DIANE Service Description) [7] has been developed within our DIANE project [8]. The project aims at developing and evaluating concepts that allow for an integrated, efficient, and effective use of resources in the form of services in ad hoc networks. Like OWL-S and WSMO, DSD is ontology based. Unlike these two, DSD describes service offers basically as sets of achievable effects. This allows for concise yet sufficiently precise descriptions.

2.1.2 Usability in a Mobile Environment Even partial automation is impossible without the usage of semantic description languages. Unfortunately, all of them require quite complex service descriptions. This poses a problem in traditional settings already. If we want to support mobile peer-to-peer environments, formed, e.g. in ad hoc networks for functionality sharing, it is even more unrealistic to expect users to put a lot of effort into devising correct descriptions of the services their devices can offer. What is needed here are means for sophisticated user support, resulting in semi-automatic generation of service requests. While there are some graphical tools available, as of now none of those is really suited for usage on a small mobile device. Clearly, future research is needed here.

2.2 Describing Service Requests

In the discussion above, we have focused our attention onto the description of service offers. Obviously, service requests need to be described, too. In this section, we take a closer look at this. First, we determine the kind of requests that will occur in a mobile environment. Then, we examine how well existing approaches can cope with these types of requests.

2.2.1 Types of Requests Let us return to the example scenario from the introduction. There, Anna (or rather her device) sent out a number of different types of requests.

First, one can distinguish requests by the type of service they need for their fulfillment: There are requests geared at information services (e.g., obtaining a map of the area), others that ask for some provision of functionality (e.g, the service offering to project a presentation on a screen), and others that enable Anna's device to do something, e.g. connect to the internet for a certain amount of time.

Second, one can distinguish between one-time and long-running requests. Anna was looking for a map at just one point in time; however, she wants to be informed throughout the conference when an attendee with similar interests is in her vicinity.

Third, one can distinguish simple requests from requests that require the aggregation of data.

Finally, the degree of context-dependency of the request differs. Typically, in mobile environments, most requests are highly context-dependent. Context includes the current location of the user, her preferences, and the device she uses, the network connection and so on.

The different types of requests do not only determine which constructs need to be offered for request formulation, but also have a strong influence on how requests are executed (see Section 4).

2.2.2 Existing Approaches In the simplest case, the service request consists of a list of keywords. As discussed below, this will not allow for automatic service matching or invocation.

All semantic service description languages allow the formulation of a service request as the description of the ideal service (OWL-S) or as an abstract description of the desired functionality (e.g. as a goal in WSMO). These approaches, however, do not allow describing how much a service may differ from this ideal service in order to be still considered appropriate. Consider, e.g., Anna, looking for an up-to-date map of her destination in a 1:10.000 scale. Assume that two services are offered: Service A offers a 2005 map with a 1:15,000 scale, whereas Service B offers a 2004 map with a 1:10,000 scale. Neither WSMO nor OWL-S allows Anna to specify in her request which of these services she would accept and which one she would prefer. In contrast, DSD offers a separate construct for capturing user preferences: Here, service requests are represented by fuzzy sets describing exactly how well any acceptable service matches the user's expectations. The usage of fuzzy sets allows capturing user preferences very precisely. Based on the fuzzy set description, a personalized matcher can be generated that is able to automatically rank service offers according to the user's preferences.

2.2.3. Usability in a Mobile Environment In order for any of the approaches to be useful in a mobile environment, a number of extensions are needed for service descriptions: First, service requests are cumbersome to come up with. It can not be expected of users to describe a semantic service request on a smart phone. Alternatives are to offer sophisticated user support or (maybe more realistically) to embed requests into applications. In this case, all that Anna would see of the services would be a menu on her smart phone, offering her choices like "get map", "register interests", "project" and so on. The application would include templates for requests that are instantiated with Anna's concrete requests, e.g., a map of Tokyo, when she enters her choice on the menu. The parameters could either be filled in manually by

Anna or derived from context information. This is a topic of our ongoing work.

3 Service Discovery and Matching

. Two things are needed to increase the ability to find appropriate services. A mechanism that allows finding service offers (service discovery) and a mechanism to efficiently and effectively compare service offers and requests (service matching)..

3.1 Service Discovery

Service offers could be published at the information source itself or at some centralized or distributed repositories [9]. The discovery mechanisms need to be able to deal with networks that change frequently and dramatically: In particular in ad-hoc networks, both network connections and resources will fluctuate significantly over time.

Service discovery mechanisms can be distinguished by whether directories are available or not. In directory-less approaches, mechanisms are provided to guide requests through the network to each individual source. The advantage here is that changes in the network do not pose a problem. The main disadvantage is the tremendous message overhead. Directory-based approaches can be divided into centralized and distributed architectures. The latter can again be subdivided into architectures for infrastructure-based systems and those that work without an infrastructure, i.e. in an ad-hoc network.

Depending on the characteristics of the mobile environment, in particular the degree to which infrastructure is available and the amount of dynamics, an appropriate solution from the alternatives above can be chosen. Overall, this question is well studied and more or less satisfactory solutions are available for each case.

3.2 Service Matching

Service matching is the process of automatically deciding whether an offered service is able to fulfill a given service request and of configuring the service appropriately in case of a match. Based on the degree to which semantic information is used in service matching, we can distinguish several classes of matchers [7].

Keyword matchers perform a "google-like" search among offers. Requests are described by a number of keywords (either freely chosen or taken from a restricted vocabulary). The matcher then determines which offers contain all or some of the required keywords. The main disadvantage of the keywords matchers is quite obvious: They are not able to produce exact results (matches). Rather, the user needs to apply more selection processing on the resulting offers to reach a suitable result which is not suitable for an automatic service selection paradigm.

Type based Matchers extend keyword search by taking a given type hierarchy into account. This allows for more precise matching than the simple keyword match, however, it still lacks precision.

Graph-based Matchers: All the service descriptions introduced above can be represented as graphs, typically as trees. Accordingly, service matching, i.e. the comparison of an offer graph and a request graph, can be regarded as a special form of (restricted) graph matching. There exist matchers based on this principle for all of the languages described here. These matchers work very well for perfect matches (i.e., offer and request are identical) and for perfect mismatches (offer and request have nothing or next to nothing in common).

Fuzzy Matchers: DSD describes service requests as fuzzy sets and can therefore determine a lot more precisely than usual graph-based matchers the exact degree of match between an offer and a request.

3.3 Service Composition

In cases of request fails where no single service represents a match to a service request, mechanisms of combining more than one service should be considered. Often, a combination of several services will be able to fulfill the offer. One can distinguish several situations in which service composition can be useful [10]:

First, a service is available that produces the desired effects. However, the preconditions of this service are not currently met. In this case, one can try to find a service that will fulfill these preconditions. Assume Anna is looking for a map in jpeg format. There may be a service offering the requested map in PDF and another one transforming from PDF to jpeg.

Second, a request may encompass several effects. For instance, Anna could be interested in the contact information of all attendees with interests similar to hers, there are services offered that provide this information for separate groups of attendees. Here, the service results need to be combined to achieve the overall goal.

Finally, additional knowledge may be needed to be able to fulfill a request. For instance, Anna may be looking for a map of Tokyo. There is a map provider that allows access via ISBN. In this case, a second service that determines the ISBN of Tokyo maps is needed.

Depending on the required combination of services a number of techniques are available to (semi-)automatically compose these services. However, this is still an active field of research.

3.4 Usability in a Mobile Environment

All the techniques described above can in principle be used in mobile environments. For service discovery, solutions that are tailored to these environments exist.

The main challenge concerning service matching is currently the lack of computing power on small mobile devices. Many matchmaking algorithms require powerful inference machines and large ontologies. This can not be provided by today's small mobile devices. We expect, that technological progress will somewhat ameliorate this problem, but that it will continue to exist to a certain degree.

Service composition is extremely important in mobile environments and still poses a number of challenging issues. Besides the open issues that exist here even in stable networks, the high dynamics in mobile environments adds complexity to this task.

4. Service Execution

Service execution should be effective and efficient. Appropriate approaches for service matching and discovery lay the basis for effective service execution. In this section, we describe how efficient execution can be achieved. The goal here is to find cost-effective execution plans. This is trivial, if just one service is needed to fulfill a certain request and the network is stable. The problem becomes challenging, however, when service composition is involved and when the environment becomes dynamic. In this case, devising the optimal execution plan is a highly complex task. This task can not be solved off-line, instead, due to the high dynamics involved, constant re-evaluation at run time is needed. Luckily, the question poses some parallels to that of query execution in distributed databases [2]. In this area, a large amount of expertise is available. However, service-orientation in mobile environments adds some new challenges: Query execution in traditional distributed database management systems has high computational complexities, especially, when introducing aggregation and nested queries [11]. These high computational complexities require considerable processing power, something that is not readily available on today's mobile devices. If we look at mobile devices, we need to consider their constraints, e.g. low battery live times, less computational power, limited available resources, and other mobility problems such as path failure, ever-changing topologies, handover and security.

In the remainder of this section, we will first give an overview of techniques from the databases community that could be adapted to our scenario. Then, we list a number of criteria that could be used to describe good execution plans. Finally, we will address the remaining challenges, i.e., the open issues that prevent us from simply applying the techniques introduced in our setting.

4.1 Query Processing

In database systems, user queries are transformed into execution plans by query optimizers [11,12]. These optimizers aim at determining execution sequences that minimize some cost function. In this section, we take a closer look at techniques geared towards dynamically changing environments:

Adaptive Query Execution Plans. In this approach, the optimized proposed query plan of the query processor is adaptive, i.e., the plan is constantly re-evaluated and can be adapted at run time. The ability of adaptation is a very important key to deal with dramatic network changes such as path failures and limited/changing available resources [12,13,14,15,16]: In the approach presented in [15], the query planning is based on dynamic programming. [17] uses an algebraic optimization model in order to generate the best schedule in terms of energy and time efficiency to maximize the number of collision free concurrent data transmissions. *Semantic Routing Trees (SRT)* are routing trees in which a participant node for given query determines if any of its neighbor/child nodes will be a data source [12]. STRs should be permanently maintained to adapt to the ever-changing mobile environment. SRTs are efficient for retrieving query results for queries over common attributes between neighbors in the routing tree and can reduce the number of nodes that must disseminate queries and forward the continuous stream of results. Other routing-based methods for adaptive processing with adaptive routing protocols such as *Eddies* are introduced in [13,14,18].

Queries involving Aggregation: These types of queries are particularly interesting since the techniques used here should be appropriate for service composition involving multiple effects.

Many schemas are introduced for this topic [12,14], in particular in the context of sensor networks. The simplest case is aggregation without any join [19,20]. The aggregation query plan is categorized into two parts: the communication and the computation part. The communication part describes how data are supposed to be collected from the network; the computation part holds instruction for the aggregation function. Depending on the amount of processing in the network different techniques can be distinguished .

Power-based Query Optimizing. Due to the limited available power for mobile devices and high complexities and power consumptions of shared processing in query optimizing and execution, optimizing query plans according to power consumption is a very important feature for query optimizers for data access processes in

mobile environments. [12], e.g., uses a low cost power optimizer that estimates many alternatives for query plans and chooses the lower power consumption plan. It also applies an event query batching to maximize power conservation.

In the following, two examples of already existing systems for query execution in uncertain ad hoc mobile networks are presented in more detail:

TelegraphCQ: The TelegraphCQ project at UC Berkeley began in 2000 with the goal of developing an adaptive dataflow architecture for supporting a wide variety of data intensive, networked applications employing many concepts of long run continuous query and on the fly processing [21]. It is using adaptive routing approaches to prepare execution plans, and is compatible with PostgreSQL query language. It represents a scalable architecture for continuous adaptive query processing for uncertain environments.

ACQP: TinyDB: The ACQP (Acquisitioned Query Processor For Sensor Networks) has been developed within the TinyDB framework. The initial release of TinyDB was developed by Kyle Stanek et al. during 2002 at Intel-Research Berkeley. Elements of the design were inspired by discussions of Berkeley TinyOS and Telegraph research projects [22]. It is a distributed query processor relying on adaptive routing techniques in order to optimize the query plan. It is using it own data retrieval language (SQL-like) with event-driven capabilities and Time-To-Life queries. TinyDb has many additional features of minimizing power consumption and events manipulation. [17] Introduces an algebraic optimization model in mobile sensor networks, it builds cost based trees in order to perform low cost-power query executions.

4.2 Performance Criteria for Optimizers

All of the optimizers introduced above aim at reducing some cost function. In traditional database systems, this cost function reflects the number of accesses to disk storage. In distributed databases, communication cost is taken into account, too. In mobile environments a number of additional factors need to be considered:

Wireless Communication costs: Cost is an important non-functional operating parameter that may be affected by many parameters of the dynamically changing mobile environment. However, while wireless connections are frequently changing, the basic principles in the relation to cost are the same as those of fixed systems [23]. Communication costs may be expressed in terms of delays, type of transmitted data, or number of connections used.

Power consumption: Some or all of the mobile devices in a mobile network may rely on batteries or other exhaustible means for their energy. For these mobile hosts, the most important system design criteria of optimization should be energy conservation [24]. Power conservation could be achieved in different concepts and on different levels of proposed architectures in service oriented environments for mobile applications: Optimizing computations of creating data requests or queries are proposed in [12]. Approaches to optimizing the middle access layer of such as routing, scheduling, and resource allocations were introduced in [24] Average consumed energy per data transmission unit is a major criterion here beside other power consumption matrices [2].

Shared processing: Mobile environments are shared processing distributed systems, applying more sophisticated shared processing paradigms similar to multi-query optimization may have some advantages especially when we consider aggregation requests.

4.3 Open Issues

In the paragraphs above, we have summarized what is there already in terms of query execution. We have also summarized criteria that should be taken into account when evaluating service execution plans in mobile environments. What we are aiming at is to apply the techniques described in Sections 4.1 to service execution planning using the criteria from Section 4.2. In this subsection, we identify what is still missing to achieve this goal and which additional factors need to be taken into consideration:

Applying Quality-of-Service (QoS) schema: Achieving desired QoS characteristics is a major optimization goal [25]. QoS schemes are used efficiently in wired networks; QoS schemas for wirelined networks cannot be applied directly into the wireless mobile networks. To support QoS, the link state information such as delay, bandwidth, and cost in the network should be available and manageable. However, managing the link state information in mobile ad-hoc networks is very difficult because the characteristics of a wireless link are ever changing with the surrounding circumstances. Furthermore, the limited resources, host's mobility and deploying a service oriented application environments add more new challenges.

While, many QoS management are proposed [1], applying these schemas in such a service oriented environment requires adaptation in order to qualitatively and quantitatively capturing mobile data access performance.

Integrating with service composition mechanisms: Automatic service composition using declarative descriptions is a key function prior to query execution. It is needed whenever the

required answers need to be collected from many partial answers. Complex and huge queries or service requests can be split into smaller ones by using efficient service composition mechanisms. Applying good composition algorithm reduces matching and searching complexities and produces easier services and query plans to be recovered. To fully exploit the benefits of service composition, a close interleaving of the composition and execution phases is needed.

Applying indexes, hashing, and data management techniques: A network can be viewed as a database. Implementing concepts that facilitate searching and execution processes from the field of database management systems is a growing topic. Unfortunately, implementing these facilitating concepts such as indexing and hashing are not trivial in uncertain mobile environments [14].

Fault tolerances recovery: When node failure or/and service unavailability occurs, some dedicated mechanisms of fault recovery should be triggered. In mobile networks, according to their characteristics, the mobile service oriented environments are supposed to have highly adapted mechanisms for service-fault recovery. Resending or re-executing queries is a very inefficient method to handle service unavailability. The broker-client approach, presented in [26], introduces a more complex algorithm depending on service composition for the same failed request.

5. Conclusions

In this paper, we have argued that service-orientation is the appropriate paradigm for mobile computing. With this architecture, it becomes possible to share functionality among mobile and stationary devices. Thus, mobile devices can appear as powerful as their stationary counterparts. To achieve this goal without needing constant user interaction, a high degree of automation is required. We have identified the key areas that need to be addressed in order for this vision to become reality: Expressive semantic descriptions of service offers and requests together with appropriate matchers are needed. Mechanisms that allow the handling of these descriptions on small mobile devices are still lacking. Service discovery is a well-established area that offers few new challenges. The main fields for research are service composition and execution. For the latter, techniques from the databases' community offer a good starting point, but cannot be applied directly. In our ongoing work we attempt to tackle these challenges.

References

[1] M. Hashem, S. Ghoniemy, M. Hamdy, "A Quality-Of-Service-Aware Genetic Algorithm for the Source Routing in Ad-Hoc Mobile Networks", ICEIS 2003, Proceedings of the 5th International Conference on Enterprise Information Systems, Angers, France, April 22-26, 2003.

[2] A. Coman, J.Sander, M.A. Nascimento, "An Analysis of Spatio-Temporal Query Processing in Sensor Networks ", 1st International Workshop on Networking Meets Databases (NetDB 2005) in conjunction with 21st ICDE, April 2005, pages 45-50.

[3] B. König-Ries, M. Klein, "Semantic Service Descriptions - Relevance for Mobile Applications, Requirements and State of the Art". Tutorial at MDM 2005, Cyprus. www2.cs.ucy.ac.cy/mdm05/MDM05-Tutorial1.html .

[4] Web Service Description Language: www.w3.org/TR/wsdl

[5] OWL-S: Web Ontology Language for Services www.w3.org/Submission/2004/07/

[6] WSMO: Web Service Modeling Ontology. www.wsmo.org

[7] M. Klein, B. König-Ries, M. Müssig, "What is needed for Semantic Service Descriptions - A Proposal for Suitable Language Constructs", the International Journal of Web and Grid Services 2005,Vol. 1, No. 3/4, pp. 328-364

[8] DIANE Project: http://hnsp.inf-bb.uni-jena.de/DIANE/en .

[9] M. Klein, B. König-Ries, P. Obreiter. „Lanes, A Lightweight Overlay for Service Discovery in Mobile Ad Hoc Network", 3rd Workshop on Applications and Services in Wireless Networks (ASWN2003). Berne, Switzerland, 2003.

[10] U. Küster, M. Stern, B. König-Ries, "A Classification of Issues and Approaches in Service Composition", First International Workshop on Engineering Service Compositions (WESC05), Amsterdam, Netherlands, 2005.

[11] R. Elmasry, S. Navathe, "Fundamentals of Database Systems", 4th Ed., Addison-Wesley, 2003.

[12] S. Madden, et al., "The Design of an Acquisitional Query Processor For Sensor Networks", ACM SIGMOD Conference, San Diego, California, USA, June 9-12, 2003.

[13] S. R. Madden, M. A. Shah, J. M. Hellerstein, "Continuously adaptive continuous queries over streams", ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, USA, 2002.

[14] V. Raman, A. Deshpande, J. M. Hellerstein, "Using State Modules for Adaptive Query Processing". Proceedings of the 19th International Conference on Data Engineering (ICDE), Bangalore, India, March 2003.

[15] F. Perich, et al., "In Reputation We Believe: Query Processing in Mobile Ad-Hoc Networks", International Conference on Mobile and Ubiquitous Systems, Boston, USA, August 2004.

[16] J. Gehrke, S. Madden. "Query Processing in Sensor Networks," IEEE Pervasive Computing, vol. 03, no. 1, January-March, 2004, Pages 46-55.

[17] V. Zadorozhny, P. K. Chrysanthis, A. Labrinidis," Algebraic Optimization of Data Delivery Patterns in Mobile Sensor Networks", 15th International Workshop on Database and Expert Systems Applications (DEXA 2004), Zaragoza, Spain, 30 August - 3 September 2004, Pages 668-672.

[18] R. Avnur, J. Hellerstein. "Eddies Continuously Adaptive Query Processing", 19th ACM SIGMOD International Conference on Management of Data Principles of Database Systems, Dallas, Texas, May 14-19, 2000.

[19] Y. Yao, J. Gehrke, "Query processing in sensor networks", Conference on Innovative Data Systems Research, CIDR, CA, USA, January 5-8, 2003.

[20] S. R. Madden, M. J. Franklin, J. M. Hellerstein, W. Hong. "Tag, A tiny aggregation service for ad-hoc sensor networks", Proceedings of the OSDI 2002, Boston, USA.

[21] S. Chandrasekaran et al., "TelegraphCQ: Continuous Dataflow Processing for an Uncertain World", CIDR, CA, USA, January 5-8, 2003.

[22] TinyDb Project: telegraph.cs.berkeley.edu/tinydb

[23] S. Chen, "Routing Support for Providing Guaranteed End-to-End Quality-of-Service", Ph.D. thesis, Engineering College, University of Illinois, 1999.

[24] K. Jamieson, "Implementation of a power-saving protocol for ad hoc wireless networks", Master thesis, Massachusetts Institute of Technology, Feb. 2002.

[25] D. Chalmers, et al., "A Survey of Quality of Service in Mobile Computing Environments", IEEE Online communication Surveys, Second Quarter 1999.

[26] D. Chakraborty, F. Perich, A. Joshi, T. W. Finin, Y. Yesha. "A Reactive Service Composition Architecture for Pervasive Computing Environments", the Proceeding of the IFIP TC6/WG6.8 ACM Conference on Personal Wireless Communications, 2002.