L-ToPSS - Push-oriented Location-based Services*

Ioana Burcea and Hans-Arno Jacobsen

Department of Electrical Engineering and
Department of Computer Science
University of Toronto
{ioana, jacobsen}@eecg.toronto.edu

Abstract. The advance in wireless networks and in positioning systems has led to a new class of mobile applications: *location-based services* (LBS). LBS offer highly personalized services to mobile users based on their locations, user profiles and static and dynamic content information. The publish/subscribe paradigm is an information dissemination model appropriate for the implementation of LBS. However, existing publish/subscribe systems do not include location information in their models. In this paper we present an extension for the publish/subscribe paradigm that can effectively support push-oriented LBS.

1 Introduction

The proliferation of mobile devices and the significant increase of data on the Internet has led to the development of a new generation of mobile applications that offer highly personalized services to mobile users. One of the most powerful ways to personalize mobile services is based on the location of the mobile client. Knowing the geographic position of the mobile user at any given time adds a new dimension to the types of offered services. Thus, *location-based services* (hereafter referred to as LBS) represent information services that exploit the location information of mobile terminals to offer highly customized information content to mobile users. Examples of LBS include: requesting the nearest business or service (e.g., the nearest restaurant or ATM), receiving proximity-based reports (e.g., traffic/weather/news reports) or proximity-based notifications (e.g., e-coupons, advertisements), payment based on proximity, tracking resources or assets, and location-based games.

Based on the information delivery method, we identify two types of LBS: push and pull-based services. Pull-based services rely on the traditional request/response paradigm, where the client browses the available services and sends a specific request to the server. The server locates the user and answers to

[0] 4th VLDB Workshop on Technologies for E-Services (TES'03)

* This research is partially funded by IBM Toronto Lab, Center for Advanced Studies, Canada

his request considering the specific location information. In push-based applications, the infrastructure autonomously pushes information to mobile terminals based on user profiles/subscriptions and their current location.

In pull-based LBS, clients have to poll the server for any information updates, which may lead to server resource contention and network overload. Furthermore, mobile systems like cell phones, PDAs and wearable computers are less suitable for browsing and query-based information retrieval due to their limited input device capabilities. The current trend toward even smaller devices will amplify this problem. Moreover, energy represents a scarce resource for mobile devices, therefore, they are better suited as passive listeners than as active tools for searching information.

The limitations presented above may be effectively addressed by a push-oriented LBS model. Push-oriented LBS offer highly personalized information to their users notifying them about events that match both their profiles / subscriptions and their current location. For example, a user can be interested in receiving information about his favorite books. He can subscribe to a targeted advertising LBS specifying his interests. While walking through the city, he can receive on his mobile device an advertisement that says his favorite book is on sale at the next corner bookstore. This kind of services would be widely accepted and used only if the information is highly relevant to the user. We believe that profile and subscription-based techniques represent effective means to achieve this goal.

The research challenges implied by this kind of applications refer to information processing. The middleware platform supporting such applications must filter information for potentially millions of users [8], given their continuously changing location information, their profiles/subscriptions and, moreover, dynamic and static content information. We argue that this kind of services can be supported extremely well by an architecture based on the publish/subscribe paradigm. The paradigm has recently gained a significant interest in the database community as support for information dissemination applications [1, 4, 6] for which other models turned out to be inadequate.

In publish/subscribe systems, clients are autonomous components that exchange information by publishing events and by subscribing to events of interest. In these systems, publishers act as information producers, while subscribers act as information consumers. A publisher usually generates a message when it wants the external world to know that a certain event has occurred. All subscribers that have previously expressed their interest in receiving such an event will be notified about it. The central component of this architecture is the event broker. This component records all subscriptions in the system. When a certain event is published, the event broker matches it against all subscriptions in the system. When the incoming event verifies a subscription, the event broker notifies the corresponding subscriber.

Due to the scalability of publish/subscribe systems in terms of number of supported clients and number of processed events per second [6], we believe that this is an extremely promising approach for push-oriented LBS.

L-ToPSS (Location-aware Toronto Publish/Subscribe System) is our research prototype that provides LBS in a push-oriented style, correlating content provider data, user profiles and continuously changing location information. In our system, content provider data is modeled as publications, while user profiles represent subscriptions. More formally, the problem that we address in this paper can be expressed as follows: given a set of mobile or stationary publishers P and their publications, a set of mobile or stationary subscribers S and their subscriptions, and continuously changing location information L about mobile entities in the system, the notifications about the matching publications are sent to the subscribers only when they are close by the corresponding publishers.

This paper is organized as follows. In the next section we briefly present some examples of LBS applications from which we derive an LBS taxonomy used to classify the problem space. Section 3 discusses the L-ToPSS research prototype and its architecture. In Section 4 we describe our experimental platform. Section 5 presents the related work. Section 6 concludes with the experience gained during the design of our model.

2 LBS - From applications to taxonomy

Generally speaking, LBS provide and deliver information to its users in a highly selective manner exploiting mobile terminal location information. Examples of applications comprise route planning applications (e.g., finding the shortest path between two locations), enhanced directory assistance (e.g., finding the nearest restaurant or ATM), location-aware advertisement (e.g., ads are targeted to consumer within a certain radius of one of the retailer's locations), safety services (e.g., tracking the location of mobile 911 callers), resource management (e.g., tracking and dispatching mobile resources) and so on. In this section we briefly describe some LBS applications that can benefit from the push-oriented style of the service.

Location-aware mobile commerce allows clients to purchase goods or services from retailers that are close to their current location. Thus, a location-based coupon service can send special offers to clients according to their profiles or preferences. Thus, clients do not have to periodically check the offers, as they are notified about them when they pass within a certain distance of the retailer's location. In this cases, the information provider has a fixed, known location, while the information requester is mobile. Moreover, the retailer's offer can be valid for an extended period of time or only at its release time.

Proximity-based alerts inform users about certain events of interest to them. For example, a user can be informed about an accident that has occurred on the highway 2 miles ahead. Based on this information, the user can request an alternative route to his destination. This kind of information is characterized by its momentary validity.

Common profile matching services allow users to be notified when a person with a compatible profile is in the area. Similarly, a buddy finder applications can notify users when a member of the family or a friend is nearby. In these scenarios, both users are mobile.

Inferring from the applications presented above, we classify the LBS based on two dimensions: the mobility scenario of the information provider (publisher) and information requester (subscriber) and the type of the publication.

Based on entities mobility, we distinguish three categories of scenarios that are presented in Table 1, where *stationary* means that the entity has a fixed, known location, while *mobile* refers to an entity that changes its location. Note that the case where both the publisher and the subscriber are stationary is not interesting from the LBS point of view. This case is addressed in traditional publish/subscribe systems.

Table 1. Mobility scenarios and application examples.

Publisher	Subscriber	Application
Stationary	Mobile	Targeted advertising
Mobile	Stationary	Airport automatic check-in
Mobile	Mobile	Friend finder

According to the life-time of the publication in the system, there are two different cases: *instantaneous* publications - the publication is valid only at its publication time, it is matched against the existing subscriptions, then discarded [1] - and *long-lived* publications - the publication inserted into the system is persistent until a corresponding delete command is performed by the publisher such that subscriptions submitted at a later point are matched against this publication. Table 2 presents examples of applications for each category.

Table 2. Publication types and application examples.

Publication type	Application
Instantaneous	Location-based games
Long-lived	Targeted Advertisement

3 L-ToPSS - System Model and Architecture

L-ToPSS (Location-aware Toronto Publish/Subscribe System) is our research prototype. The issue of obtaining location updates of mobile clients is independent of the system behavior and it is beyond the scope of our research [16]. We assume that the system can periodically receive location information about its users as *(latitude, longitude, altitude)* coordinates. With the advances in positioning technologies, we believe that getting location updates does not represent an issue anymore.

[1] In current publish/subscribe systems, the publications are always instantaneous.

3.1 System Model

The main component of the system is the filtering engine that matches the publications against the subscriptions in the system. In the model we propose, the publications describe real life objects, such as books that, for example, can be characterized by title, author, and edition. This type of information can be represented by semi-structured data used in traditional publish/subscribe systems. In our system, the publication is expressed as a list of attribute-value pairs. The formal representation of a publication is given by the following expression: $\{(a_1, val_1), (a_2, val_2), \ldots, (a_n, val_n)\}$. For the book example presented above, the publication can be expressed as:

{*(title, "Location-based service"), (author, "H-A.Jacobsen"), (edition, 2003)*}.

The subscriptions describe user interests or user profiles. In our system, subscriptions are represented as conjunctions of simple predicates. Each predicate expresses a value constraint for an attribute name. For example, the predicate ($edition > 2000$) restricts the value of the attribute *"edition"* to a value greater than 2000. In a formal description, a simple predicate is represented as *(attribute_name relational_operator value)*. A predicate *(a rel_op val)* is matched by an attribute-value pair *(a', val')* if and only if the attribute names are identical *(a = a')* and the *(a rel_op val)* boolean relation is true. In our system, a subscription s is matched by a publication p if and only if all its predicates are matched by some pair in p.

If either the publisher or the subscriber is stationary, we assume that the publication or the subscription, respectively, is associated with the fixed location of the corresponding entity. The location information is expressed as *(latitude, longitude, altitude)* coordinates. Similarly, when an entity is mobile, the information it produces contains the *Mobile Identification Number* (MIN) - a unique identifier of the mobile device.

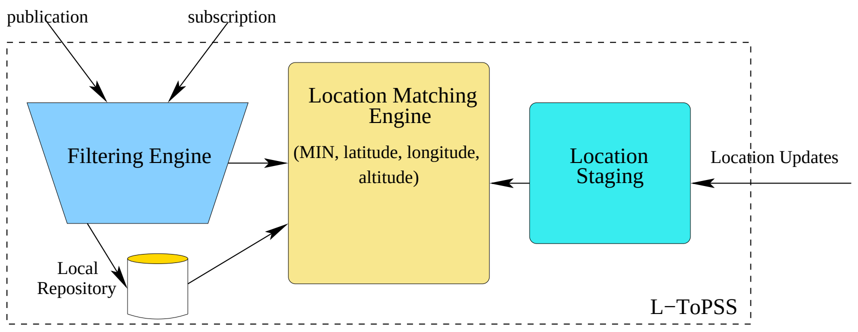

Fig. 1. L-ToPSS System Architecture

3.2 System Architecture

The system architecture is depicted in Figure 1. First, we explain how the system works for the stationary publisher-mobile subscriber case. Then, we argue that the mobile publisher-stationary subscriber scenario can be treated symmetrically. Finally, we present the mobile publisher-mobile subscriber case.

Both subscriptions and publications are sent to the filtering engine. When a subscription is matched by a publication, a *location constraint* that contains the *MIN* of the subscriber and the *(latitude, longitude, altitude)* coordinates of the publisher is sent to the location matching engine. This component stores the location constraints, as well as the associations with the subscriptions and the publications that have generated them.

If the publication is a long-lived one, it is stored in a local repository. In this way, subscriptions entering the system will be matched against the existing publications in the repository. For each match, a location constraint is created in the same way as explained above and then, it is sent to the location matching engine. The location constraint is kept in the location matching engine as long as the publication exists in the system. Conversely, if the publication is instantaneous, it is not stored in the system. The instantaneous publication and the location constraints that it produces are discarded after a period of time equal to the duration of a cycle of location updates (i.e., the time needed for receiving and processing the location updates for all connected clients). This means that in order to receive notifications about instantaneous publications, interested subscribers have to be in the area of the publisher at the moment when the publication is issued.

The system periodically receives updates of users' location. This information is processed in the location staging component. Each location information is represented as a *(user_MIN, current_latitude, current_longitude, current_altitude)* tuple. This tuple is forwarded to the location matching engine that matches it against the location constraints in the system. A tuple *(user_MIN, current_latitude, current_longitude, current_altitude)* matches a location constraint *(MIN, latitude, longitude, altitude)* if and only if *MIN = user_MIN* and the distance[2] between the two points determined by *(current_latitude, current_longitude, current_altitude)* and *(latitude, longitude, altitude)* does not exceed a certain value. If a location constraint is matched, this means that the corresponding subscriber is close to a point of interest for him. Therefore, the system will send a notification to the user about the publication associated with the location constraint. The notification is sent to the mobile device identified by the *MIN*. After the notification is sent, the location constraint is deleted. In this way, the user will be notified at most once about a publication, avoiding sending the user the same piece of information over again.

The mobile publisher-stationary subscriber case can be modeled symmetrically. In this case, the static location is associated with the subscription, while the *MIN* is contained in the publication. The system processes the information in

[2] The distance can be expressed as a function defined by the subscriber

the same way as in the previous case. In this scenario, the stationary subscriber will be notified when the publisher comes nearby.

For the mobile publisher-mobile subscriber case, each entity has associated a MIN: MIN_{pub} and MIN_{sub}. In this case, the location constraint contains only the (MIN_{pub}, MIN_{sub}) tuple and it is associated with the corresponding publication and subscription. For each MIN that appears in the location constraints, the system stores the last location update and the timestamp when it was received. The location matching proceeds as follows. When a location update $(MIN_1, latitude, longitude, altitude)$ enters the system, the corresponding location information and the timestamp are updated. Moreover, for all the location constraints (MIN_1, MIN_2), the system checks if the last location received for MIN_2 is close to that of MIN_1 and also if the timestamps are close in time. If this is the case, the appropriate subscriber is notified about the publication associated with the location constraint.

Both the publisher and the subscriber can retrieve their publication or subscription, respectively. When a publication or a subscription is deleted, all the corresponding location constraints have to be deleted.

3.3 Information Processing

In this subsection we present the sequence of operations that correspond to information processing in the system, i.e., subscriptions, publications, location updates. The operations are considered in the stationary publisher - mobile subscriber case.

Insert publication: insert a long-lived publication in the system
Input: publication P, location of the stationary publisher $(lat_p, long_p, alt_p)$

- store the publication P in the local repository
- match the publication P against the subscriptions in the system and retrieve the set of all matching subscriptions $\{(MIN_{k_1},\ S_{k_1}),\ (MIN_{k_2},\ S_{k_2}),\ \ldots,\ (MIN_{k_m},\ S_{k_m})\}$
- create the corresponding location constraints $\{(MIN_{k_1},\ S_{k_1},\ \mathtt{P},\ lat_p,\ long_p,\ alt_p),\ (MIN_{k_2},\ S_{k_2},\ \mathtt{P},\ lat_p,\ long_p.\ alt_p),\ \ldots,\ (MIN_{k_m},\ S_{k_m},\ \mathtt{P},\ lat_p,\ long_p,\ alt_p)\}$ and send them to the location matching engine

Delete publication: delete a publication from the system
Input: publication P

- delete P from the local repository
- delete all location constraints that are associated with P from the location matching engine

Match publication: send an instantaneous publication to the system
Input: publication P, location of the stationary publisher $(lat_p, long_p, alt_p)$

- match the publication P against the subscriptions in the system and retrieve the set of all matching subscriptions $\{(MIN_{k_1},\ S_{k_1}),\ (MIN_{k_2},\ S_{k_2}),\ \ldots,\ (MIN_{k_m},\ S_{k_m})\}$

- create the corresponding location constraints $\{$`(`MIN_{k_1}`, `S_{k_1}`, P, `lat_p`, `$long_p$`, `alt_p`), (`MIN_{k_2}`, `S_{k_2}`, P, `lat_p`, `$long_p$`, `alt_p`), ..., (`MIN_{k_m}`, `S_{k_m}`, P, `lat_p`, `$long_p$`, `alt_p`)`$\}$ and send them to the location matching engine specifying that the publication is instantaneous

Insert subscription: insert a subscription in the system
Input: subscription `S`, mobile identification number `MIN`

- store the subscription in the filtering engine
- query the local repository and retrieve the set of all matching publications $\{$`(`P_{l_1}`, `lat_{l_1}`, `$long_{l_1}$`, `alt_{l_1}`), (`P_{l_2}`, `lat_{l_2}`, `$long_{l_2}$`, `alt_{l_2}`), ..., (`P_{l_n}`, `lat_{l_n}`, `$long_{l_n}$`, `alt_{l_n}`)`$\}$
- create the corresponding set of location constraints $\{$`(MIN, S, `P_{l_1}`, `lat_{l_1}`, `$long_{l_1}$`, `alt_{l_1}`), (MIN, S, `P_{l_2}`, `lat_{l_2}`, `$long_{l_2}$`, `alt_{l_2}`), ..., (MIN, S, `P_{l_n}`, `lat_{l_n}`, `$long_{l_n}$`, `alt_{l_n}`)`$\}$ and send the location constraints to the location matching engine

Delete subscription: delete a subscription from the system
Input: subscription `S`

- delete the subscription `S` from the filtering engine
- delete all location constraints that are associated with `S`

Match location update: match a location update against the location constraints in the system
Input: mobile identification number `MIN` and the corresponding location information `(`$lat_{MIN}, long_{MIN}, alt_{MIN}$`)`

- retrieve all location constraints that contain `MIN`: $\{$`(MIN, `S_{k_1}`, `P_{l_1}`, `lat_{l_1}`, `$long_{l_1}$`, `alt_{l_1}`), (MIN, `S_{k_2}`, `P_{l_2}`, `P_{lat_2}`, `$long_{l_2}$`, `alt_{l_2}`), ..., (MIN, `S_{k_r}`, `P_{l_r}`, `lat_{l_r}`, `$long_{l_r}$`, `alt_{l_r}`)`$\}$
- for i := 1 to r do
  - if the distance between `(`lat_{MIN}`, `$long_{MIN}$`, `alt_{MIN}`)` and `(`lat_{l_i}`, `$long_{l_i}$`, `alt_{l_i}`)` is within a certain value, send P_{l_i} to the mobile device identified by `MIN` and delete the location constraint `(MIN, `S_{k_i}`, `P_{l_i}`, `lat_{l_i}`, `$long_{l_i}$`, `alt_{l_i}`)`
  - else if P_{l_i} is instantaneous, then delete the location constraint `(MIN, `S_{k_i}`, `P_{l_i}`, `lat_{l_i}`, `$long_{l_i}$`, `alt_{l_i}`)`

3.4 Modeling Applications with L-ToPSS

In this subsection we present three examples of applications and show how they can be modeled with our system: the targeted advertising application we referred to earlier in the paper, an application for automatic airport check-in and a friend finder application.

Targeted Advertising Application
This scenario demonstrates the deployment of an LBS application that informs

its users about objects of interests that are close to their current position. The objects of interest are matched according to user profiles. Thus, users are notified about discounts or sale for items that correspond to their interests and are available in their proximity. In this type of application, mobile clients act as subscribers in our model, while content providers represent stationary publishers. User profiles represent subscriptions, while information made available by the content providers is inserted as publications in the system. Thus, when a mobile user submit its profile to the system, the method *insert subscription* is called with the appropriate parameters: the MIN of the client and the profile expressed as a subscription (see Section 3.1, for details). Similarly, the information made available by the content provider is inserted into the system using the method *insert publication* or *match publication* depending on whether the publication is instantaneous or a long-lived one. The location updates of the mobile client will be sent to the system by the *match location update* method.

Automatic Airport Check-in
This application addresses the following scenario. A client buys a plane ticket. When he arrives at the airport, the check-in system detects his presence at the airport and performs automatic check-in for the client. This application can be supported in our system using the mobile publisher-stationary subscriber approach. The mobile client acts as a publisher. When he buys the plane ticket, the ID of the flight and the MIN of the user's mobile device are inserted into the system as a publication, using the *insert publication* method. The airport check-in system acts as a subscriber: when the check-in procedure for a certain flight has to start, it subscribes to the ID of that particular flight, using the method *insert subscription.* This subscription will match all publications that contain that particular flight ID. In this way, a set of location constraints are created and sent to the location matching engine. When the client arrives at the airport, the corresponding location update will match the location constraint of that client; thus, the check-in system (i.e., the subscriber) is notified and it can perform automatic check-in for the client.

Friend Finder Application
This service allows mobile users to specify a member of the family or a fried about whom they would like to be notified when he is in the same area. This application follows the mobile publisher-mobile subscriber scenario. Practically, this scenario coordinates the positions of the two mobile users. Each mobile user has associated a MIN of the mobile device. When one user subscribes to the service for receiving notifications when his buddy is nearby, the corresponding location constraint is created (i.e., (MIN_1, MIN_2)). The location constraint will be matched only when the two mobile users are in the same area and the appropriate notification is sent to the subscriber.

These three applications presented above cover all mobility scenarios described in Section 2. All other applications that follows a particular scenario can be modeled in a similar way.

4 L-ToPSS - Experimental Platform

Our experimental platform is depicted in Figure 2. In order to experiment with our system, we use two IBM tools designed to facilitate LBS testing: City Simulator [11] and Location Transponder [14] . The City Simulator tool produces trace-files that simulate the motion of up to 1 million people. The trace-files contain timestamped data records representing coordinate positions of mobile objects (e.g., cell phone users). The Location Transponder takes as input the trace-files produced by the City Simulator and transmits the data to a receiving LBS application/server using HTTP requests, SOAP calls, UDP datagrams or a user-provided custom method. Thus, the Location Transponder will provide the location information updates to our system.

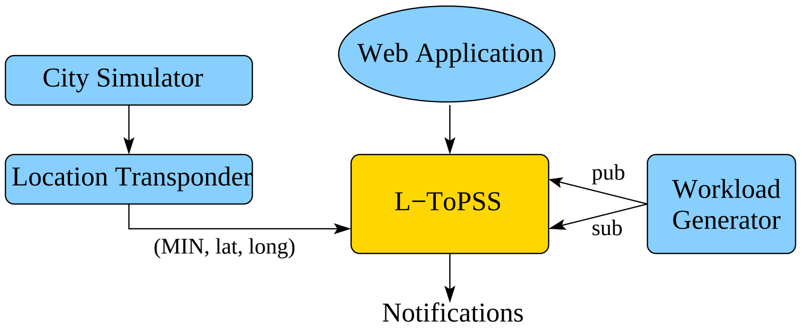

Fig. 2. L-ToPSS Experimental Platform

To experiment with the applications presented above, we develop a web application for client registration/profile input as well as for specifying publications. In order to simulate the activity of a real-life system, we use a workload generator which sends subscriptions and publications randomly into the system. The location matching engine and the repository used for storing publications are implemented using fastDB [7]

Table 3. ToPSS matching engine: matching performance for one event and different number of subscriptions.

# subs	matching time (ms)
1000	0.10
5000	0.27
10000	0.63
50000	3.85

# subs	matching time (ms)
100000	8.10
500000	79.00
1000000	190.70
5000000	1056.00

We believe that the main bottleneck in this architecture is the filtering engine. Our filtering engine is based on the ToPSS high-throughput matching kernel. The kernel implements a variety of counting algorithms [3]. We have evaluated its performance on a dual-CPU Pentium II workstation with an i686 CPU at 900MHz and 1.5GB RAM operating under Linux RedHat 7.2. Table 3 summarizes the experimental results for matching an event against different number of subscriptions. As the matching of location updates and the matching of subscriptions represent queries against tables, they are not interesting from an experimental point of view. Therefore, we do not include results about them.

5 Related Work

The work in this paper is part of the Toronto Publish/Subscribe System (ToPSS) project of the Middleware Systems Research Group (MSRG). Other recent and ongoing projects in the MSRG include Approximate ToPSS [13], Semantic ToPSS-[15], and Subject Spaces [12].

A lot of research has been devoted to developing publish/subscribe systems [1, 2, 4, 6]. Only recently, some research addresses location information in publish/subscribe systems. Chen et al [5] present an architecture for filtering spatial events (i.e., location updates). However, they do not support in their model the correlation of user profiles, content provider data and location information. [9] presents a multidimensional model for representing the information in LBS. Other topics of research related to our work are querying location dependent data, spatial databases and moving objects [10, 17]. These topics address issues related to data management, data indexing and representation. Publish/subscribe systems solve a problem inverse to the database query processing: subscriptions represent queries, while publications represent data items; usually, publish/subscribe systems filter data items against queries. [10] contains an approach similar to the publish/subscribe model: in contrast with traditional approach of building indexes on moving objects, they introduce query indexing (queries remain active for a long period of time, while moving objects change their position continuously). In industry, content and location providers have implemented different aspects of LBS, but they focus on pull-oriented services.

6 Conclusions

In this paper we present an extended model for a publish/subscribe system that is location-aware. While the results presented in the paper are only preliminary, we consider that the model introduced can effectively support location based services in a push-oriented style. During the design of the model, we learned that there are cases when the publications need to be persistent in the system. Moreover, the location information can be processed independently from the matching between publications and subscriptions. In this way, we avoid updating subscriptions with each location change of the subscriber.

References

1. Marcos Kawazoe Aguilera, Robert E. Strom, Daniel C. Sturman, Mark Astley, and Tushar Deepak Chandra. Matching events in a content-based subscription system. In *Symposium on Principles of Distributed Computing*, pages 53–61, 1999.
2. Mehmet Altinel and Michael J. Franklin. Efficient filtering of xml documents for selective dissemination of information. In *Proceedings of the 26th VLDB Conference*, 2000.
3. Ghazaleh Ashayer, Hubert Ka Yau Leung, and H.-Arno Jacobsen. Predicate matching and subscription matching in publish/subscribe systems. In *Workshop on Distributed Event-based Systems*, Vienna, Austria, 2-5 July 2002.
4. Antonio Carzaniga, David S. Rosenblum, and Alexander L Wolf. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems*, 19(3):332–383, August 2001.
5. Xiaoyan Chen, Ying Chen, and Fangyan Rao. An efficient spatial publish/subscribe system for intelligent location-based services. In *Workshop on Distributed Event-based Systems*, San Diego, California, 8 June 2003.
6. Francoise Fabret, H.-Arno Jacobesen, Francois Llirbat, Joao Pereira, Kenneth Ross, and Dennis Shasha. Filtering algorithms and implementation for very fast publish/subscribe systems. In *SIGMOD Conference*, 2001.
7. Fastdb. "http://www.ispras.ru/~knizhnik/fastdb.html".
8. H.-Arno Jacobsen. Middleware services for selective and location-based information dissemination in mobile wireless networks. In *Workshop on Middleware for Mobile Computing, Middleware 2001*, Heidelberg, Germany, 12-16 November 2001.
9. Christian S. Jensen, Augustas Kligys, Torben Bach Pedersen, and Igor Timko. Multidimensional data modeling for location-based services. In *The tenth ACM international symposium on Advances in geographic information systems*, Virginia, Germany, 2002.
10. Dmitri V. Kalashnikov, Sunil Prabhakar, Susanne Hambrusch, and Walid Aref. Efficient evaluation of continuous range queries on moving objects. In *DEXA 2002, Proc. of the 13th International Conference and Workshop on Database and Expert Systems Applications*, Aix en Provence, France, September 2–6 2002.
11. James Kaufman, Jussi Myllymaki, and Jared Jackson. City simulator. November 2001. IBM alphaWorks emerging technologies toolkit, http://www.alphaworks.ibm.com/tech/citysimulator.
12. Hubert Leung and H.-Arno Jacobsen. Subject spaces: A state-persistent model for publish/subscribe systems. In *Computer Science Research Group Technical Report CRSG-459*, University of Toronto, September 2002.
13. Haifeng Liu and H.-Arno Jacobsen. A-topss - a publish/subscribe system supporting approximate matching. In *Very Large Databases (VLDB'02)*, University of Toronto, August 2002.
14. Jussi Myllymaki and James Kaufman. Location transponder. April 2002. IBM alphaWorks emerging technologies toolkit, http://www.alphaworks.ibm.com/tech/transponder.
15. Milenko Petrovic, Ioana Burcea, and H.-Arno Jacobsen. S-topss - a semantic publish/subscribe system. In *Very Large Databases (VLDB'03)*, Berlin, Germany, September 2003.
16. Y.Zhao. Standardization of mobile phone positioning for 3g systems. IEEE Communicantions Magazine, July 2002.
17. Jun Zhang, Manli Zhu, Dimitris Papadias, Yufei Tao, and Dik Lun Lee. Location-based spatial queries. In *SIGMOD Conference, to appear*, 2003.