Distributed Tuplespace and Location Management - an Integrated Perspective using Bluetooth

Anders Fongen and Christian Larsen
The Norwegian School of Information Technology
Email: anders@fongen.no

Gheorghita Ghinea and Simon J E Taylor
Brunel University, Uxbridge, UK

Abstract— **Location based or "context aware" computing is becoming increasingly recognized as a vital part of a mobile computing environment. As a consequence, the need for location-management middleware is widely recognized and actively researched.**

Location management is frequently offered to the application through an API where the location is given in the form of coordinates. It is the opinion of the authors that a localization API should offer localized data (e.g. direction to the nearest pharmacy) directly through a transparent and integrated API.

Our proposed middleware for location and context management is built on top of *Mobispace*. Mobispace is a distributed tuplespace made for J2me units where replication between local replicas takes place with a central server (over GPRS) or with other mobile units (using Bluetooth). Since a Bluetooth connection indicates physical proximity to another node, a set of stationary nodes may distribute locality information over Bluetooth connections, and this information may be retrieved through the ordinary tuplespace API.

I. Introduction

Middleware for location based computing is typically found behind location APIs like JSR-179, through which the client program can inquire about its own position. It is then the responsibility of the application to retrieve the necessary localized information. This operation may involve transformation of coordinates to retrieval parameters which potentially is a complicated process.

A more straightforward approach to the retrieval of localized information is needed. In this paper, a location middleware is offered as an integral part of a distributed tuplespace system. Localized tuples (i.e. tuples containing local information) are retrieved from the tuplespace as any other tuple.

The proposed implementation of the location service is based on a distributed tuplespace for J2me (Java 2 Micro Edition) called *Mobispace*[1], in which mobile nodes update each other's local store over Bluetooth/GPRS connections. It is thus possible to configure "fixed" nodes with a Bluetooth adapter working as "beacons" so that other nodes within radio range will know the name of the "area" they are in, and on the basis of this information fetch localized tuples from the local store. The focus of this paper is to provide detailed information on the principles of this mechanism.

II. An overview of Mobispace

Mobispace is an implementation of the tuplespace model for coordination, communication and storage, also known as *Linda*[2]. A large body of knowledge has been established on how to design distributed applications over the tuplespace abstraction (e.g. [3]).

The Mobispace system is designed for mobile applications. It utilizes the limited resources present in a mobile unit and is designed for connection interruption of unknown length. For portability reasons, the Java 2 Micro Edition (J2me) platform has been chosen for the implementation for portability reasons.

The typical communication facilities for a J2me device is a GPRS/GSM service which offers HTTP connections through the Internet, and/or a Bluetooth device offering short-distance communication with other mobile units (or possibly a larger computer). Mobispace uses a distributed and replicated tuplespace employing replication methods that exploits a combination of these communication facilities.

The attractiveness of the Mobispace is that it offers a familiar and flexible programming model with a high abstraction level to developers of mobile systems. The loosely coupled coordination and indirect interactions offered by the tuplespace model fits well with the dynamic environment of mobile systems.

Mobispace features are:

- Primary-based replication based on a central (primary) server connected to mobile (secondary) nodes through a GPRS/GSM service (or any service that can offer an IP connection)
- p2p-based replication between secondary nodes based on Bluetooth communication
- Secondary nodes express their tuplet selection criteria during replication through a set of templates called an *interest profile*
- Open protocols (XML, HTTP, RFCOMM) for interoperability with non-J2me agents. Secondary nodes can run on any platform and in any language
- Unknown and dynamic number of secondary nodes
- Straightforward ordering and synchronization semantics

III. The principles of tuplespace programming

The programming model known as "tuplespace" was proposed by Gelernter in 1985 [2] as a combination of an associative shared storage mechanism and synchronized retrieval operations in a model called *Linda*. Today there are two major implementations of tuplespace in a Java environment:

1-4244-0523-8/07/$20.00 ©2007 IEEE

JavaSpaces from Sun Microsystems [4], [5] and IBM TSpaces [6].

The basic data structure used in the tuplespace is the *tuple*, which is an ordered set of *fields*. Tuples may be written to the tuplespace, after which they are available for retrieval by any client of the tuplespace. The original tuplespace model makes a clear distinction between consuming and non-consuming retrieval operations: A consuming retrieval operation is an atomic read-delete operation, so that it guarantees that only one client retrieves the tuple. A non-consuming retrieval operation returns a tuple without affecting its existence. A tuple does not need any unique fields in the sense of a primary key.

Retrieval of tuplets is done through the use of a *template* parameter. The retrieval operation selects a set of tuples *matching* the template, and one or all of the matching tuples are returned to the caller. A template resembles a tuple by its ordered set of fields, but some of the fields may be "wildcards" i.e. they have no defined value. A tuple matches a template if all these conditions are met:

- they have the same arity (number of fields),
- the fields of the template and the tuple have pair-wise the same value and type. Wildcard fields in the template matches any field value in the tuple.

The original Linda model uses typeless wildcards, and the JavaSpaces implementation follows this principle. IBM's TSpaces, on the other hand, uses *typed wildcards*, in which the type of the wildcard is checked against the type of the tuple field in an object-oriented fashion.

Neither the Linda model nor JavaSpaces offer any defined order of retrieved tuplets. TSpaces offers 'FIFO' ordering as a configuration option. In JavaSpaces, any ordering requirements is left to the application which must implement a sequence number scheme in the tuple design whenever needed.

IV. Distributed tuplespaces

Both JavaSpaces and TSpaces implement their services based on a central server. A central server facilitates consistency and transactional semantics while at the same time creating a scalability bottleneck and a single point of failure. Also, a central server most often requires permanent connectivity between the client and the server. Therefore, several distributed tuplespaces have been proposed: Patterson [7] has presented a fault-tolerant distributed design which requires high availability of network resources. The LIME system (Linda in a Mobile Environment) [8] offers a platform for mobile agents which bring a small tuplespace with them as they migrate and make them accessible to other agents residing on the same host. The SwarmLinda system [9] offers a mechanism for distributed clustering of tuples in a p2p environment and claims to be highly scalable. No distributed tuplespace implementation for the J2me environment has been reported.

In order to maintain the transactional semantics of a tuplespace system the clients need (in practice) to be permanently connected to the server, so the state oriented operations between the nodes can be effectively conducted. A consuming read, for instance, will require a lock on the same tuple in all replica in order to provide a guarantee that the tuple is taken by only one client, and such a stateful distributed operation requires high availability of network resources.

A distributed tuplespace designed for an occationally connected environment requires a reformulation of the transactional semantics. A scheme that allows for relaxed coordination between nodes is required. Ordering semantics combined with lazy replication appear to be useful elements of such a scheme.

A. Ordering and consistency semantics

The correctness of a replicated storage system relies on the ordering of write operations being passed across the network. If two replica receive write operations in different order, they may end up in different (inconsistent) states.

A system where all replica receive the results of write operations in the same order is called *sequentially consistent*. A more relaxed requirement is that all nodes should receive *causally related* write operations in the same order, in which case the system is *causally consistent*. The corresponding ordering requirement is called *causal ordering*. Mobispace offers causal ordering semantics.

Although considerable effort have gone into semantic definitions of tuplespace based coordination models, e.g. [10], there has been no reports on the semantics of tuple ordering.

The deeper details of the Mobispace architecture has been presented in [1]. The text will now proceed with a discussion on how to use Bluetooth based replication for location management purposes.

V. Bluetooth as a basis for location management

On top of the current Mobispace configuration, location management comes almost for free. A secondary node must for this purpose be equipped with a Bluetooth adapter. Two Bluetooth units can "discover" each other and inquire about the other node's name and available sevices, and then connect for transport of data.

A stable (non-moving) secondary node can be configured to act as a *beacon*, and the area within radio range of its Bluetooth adapter is called a *zone*. Other nodes within radio range will pick up its "friendly name" as a designation of its location. The "zone designator" is used as a field to construct the template being used for retrieval of *localized tuples* i.e. tuples which are valid only in this zone.

Several research projects attempt to utilize Bluetooth hardware for purposes of location management [11], [12], [13], [14], [15]. Although not designed with instant device discovery in mind, Bluetooth has interest due to its wide deployment in mobile units and well-established API.

Figure 1 shows an example on how a Mobispace network can be configured for location management purposes. Three secondary nodes are deployed as beacons on ordinary PCs representing the three zones "MainLobby", "Cafeteria" and

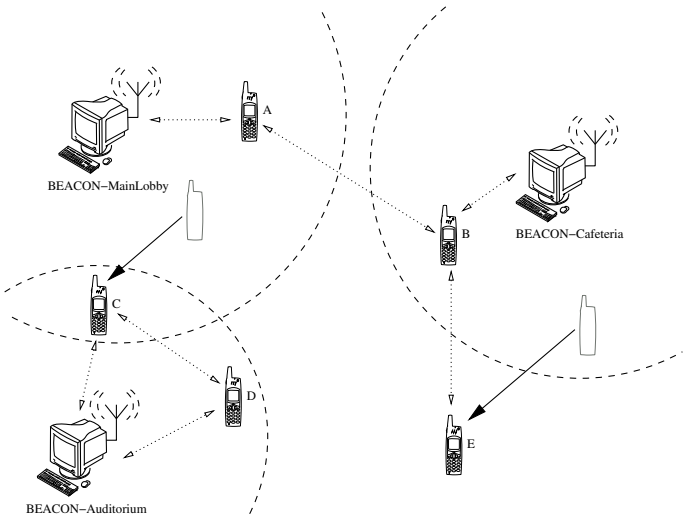

Fig. 1. Configuration and positioning of localization hardware

"Auditorium". They are identified as beacons by other nodes by naming convention.

The mobile nodes that are within radio range of one beacon (nodes A, B and D) will have discovered the beacon and set up their tuplespace retrieval templates accordingly. The mobile nodes C and E have recently moved as shown with solid-line arrows on the figure. Node E is now outside the range of all beacons, but retain its association with the zone "Cafeteria" until it eventually moves within radio range of another beacon. Node C has moved within radio range of the beacon representing the zone "Auditorium", but since it still hears the old beacon ("MainLobby") it will still be associated with this zone.

The dashed-line arrows show a selection of secondary replication links. They are included to show that the secondary replication takes place fully independent of the associations of nodes to zones: B and A replicate while in different zones, and C replicates with a beacon which it is not associated with.

Although not shown on the figure, secondary nodes (mobile and beacons) are optionally conducting primary replication with the primary server; beacons are likely to use wired connections for this purpose, whilst mobile nodes e.g. use GPRS.

The selection of localized tuples represents a process independent from the replication strategy. In other words, the localized tuples may already be present in a node as it enters the zone. The localized tuples are replicated between the nodes in the same fashion as any other tuple, which means that the *interest profile* must be set accordingly for the mobile node to receive localized tuples.

A. The Design of a localized tuple

A new tuple field data type has been introduced for the purpose of location management, the *Location*. Due to the type matching of templates and tuples, no localized tuple will be returned to a client unless the template has a field of this data type. A distinct data type for this purpose thus strengthens the separation between localized and ordinary tuples.

For the current state of this project, localized tuples are in the form of (key,value) pairs. A localized tuple has the following design:

$$localizedTuple = \{Location(zone), String(key), String(value)\}$$

which means that it contains of three fields, the first one being of type *Location*, the two following of type *String*. The value of the first field indicates the zone designation that the tuple belongs to.

The retrieval of localized tuples which belong to a particular zone will use the zone designation and value key as fields in the template parameter:

$$localizedTemplate = \{Location(zone), String(key), String(wildcard)\}$$

The management of localized tuples (creation and deletions) may be given to any node in the system, but the best solution is to leave this task to the beacons itself or a central coordinator.

B. User-centric localized tuples

In addition to these "zone-centric" localized tuples there exist also localized tuples that do not describe properties of locations, but of *users*. User-centric localized tuples are used to describe the whereabouts of user/nodes[1] so that questions like: "In which zone is Christian?" or "Who is in the *Cafeteria* zone?" may be answered.

The design of a user-centric localized tuple involves the same structure as before but involves a "null" zone designation which indicates that it is valid in all zones.

$$localizedUserTuple = \{Location(null), String(user), String(zone)\}$$

The management of user-centric localized tuples is done automatically by the Mobispace middleware. As soon as a node comes within radio range of a beacon and establishes a link with it, the Mobispace software of the mobile node will remove the tuple containing its former location and replace it with an updated value (with the designation of the new zone). This information (both the tuple deletion and the new tuple) will eventually propagate to all nodes through replication sessions. Causal tuple ordering will assure that the information is received in the correct order.

The retrieval of localized tuples which describe the location of a particular user will use a template like:

$$localizedUserTemplate = \{Location(null), String(user), String(wildcard)\}$$

[1]We assume that a node represents a user, and thus the location of nodes reveals the location of a person.

The retrieval of user-centric localized tuples which belongs to a particular zone will use the zone designation and value key as fields in the template parameter:

$$\begin{aligned} localizedZoneTemplate &= \{Location(wildcard), \\ &\quad String(wildcard), \\ &\quad String(zone)\} \end{aligned}$$

Questions like "who is in zone Y" or "who is in all zones" is answered by applying the *localizedZoneTemplate* to a retrieval operation.

C. Zones larger than the radio range

A mobile node picks up the zone designation as it discovers a beacon, and keeps that designation as "its" until another beacon is heard. Consequently, a node belongs to a zone from the moment it discovers one beacon until it discovers the next (as indicated on Figure 1). This condition of the system can be exploited in order to have zones which are larger than the radio range of a small Bluetooth beacon: A beacon may be placed e.g. in the entrance of a building in order to have one zone for the entire building, since every mobile node present in the building has to pass the beacon in the entrance. Efficient physical placement of beacons should therefore not only consider the propagation of radio waves, but also the movement patterns of the users.

D. Scalability and responsiveness

The described form of location management depends on the responsiveness of the underlying communication services. The example just mentioned with a beacon in the entrance of the building requires that a node quickly detects a that a beacon has come inside radio range and quickly establishes the identity of the new zone. Also for application where it is necessary to keep a trace of movements in the form of a sequence of zone designations, it is important that this process completes before the user moves out of radio range again. In other words, the size of the Bluetooth "cell" should be large enough so that even a user in constant movement should be able to establish the zone identity before it moves on. It also becomes necessary to consider scalability issues: There is an upper limit on how many mobile nodes that can enter the building at the same time so that everyone discovers the beacon.

Bluetooth technology is not particulary designed for quick link establishment. Bruno and Delmastro [12] show how the discovery time (equivalent to "link setup") forms a two-lobed probability distribution with peaks at approx. 0.5 sec and 3.0 sec. The two-lobed distribution is due to the random selection of frequency sequences in the bluetooth nodes. Their report also shows that in piconets with 7 nodes or less, half of the nodes will be found within 0.8 sec. After 3.3 sec all nodes are found by the inquiring master, even in configurations with as many as 15 nodes.

After a device discovery phase, the inquiring node will normally initiate a *Service Discovery* phase in order to find out if the detected nodes belong to the same application. The outcome of a Service Discovery is a URL which can be used to connect to the announced service in another node.

Whereas the Device Discovery phase is mandatory in order to establish a Bluetooth link between two nodes, the Service Discovery phase is not. Optimization of the discovery phases can thus use several techniques:

- Bypass the Service Discovery Protocol (SDP). One purpose of the SDP is to determine the URL necessary to connect to a particular service of a Bluetooth node. This URL will change each time the node restarts its service. Our choice has been to put the URL as a tuple in tuplespace when the service is started, so that a client may look for the URL in tuplespace rather than doing a SDP inquiry. It there is no URL in tuplespace, or the given URL does not work, the node initiates a SDP inquiry. Experimental evaluation estimates the effect of this technique to be approximately 1.1 second.
- Don't let beacons do Device Discovery (DD). During Device Discovery a node cannot be discovered or receive connections from others, so a beacon increases its availability to others if it refrains from DD. Mobile nodes will know that this is a beacon (by convention in its Friendly name) and connect to it. Since two beacons are never expected to connect to each other, this scheme works without problems.

The next section of the paper will present a strategy for an improved discovery process, which will lessen the requirement for frequent inquiries in a mobile node.

VI. Diffusion of location information

Due to a number of reasons discussed earlier in the paper, a bluetooth device discovers another device with a probability < 1: Configuration parameters, existing connections, implementation restriction, inquiry interval, speed of movement and scalability issues may contribute to a reduced probability for discovery.

Rather than to fight against all these factors, our strategy has been employed to "relax" the discovery requirements while at the same time increase the probability of successful zone discovery.

The strategy has been called "Positioning by diffusion" and has been devised my Spratt [16]. The idea is that positioning information may be passed between mobile nodes as well as from a stationary beacon. Given that a mobile node can only move with an upper speed limit, a time-based calculation can estimate the accuracy of a position at any given instant. The possible positions of a mobile node lies within a circle where the center is the position of the last heard beacon and the radius is the time elapsed since it was heard multiplied with the maximum speed of movement.

A mobile node who gets informed from several other mobile nodes about their estimated positions can estimate its own position as the intersection between several circles (one circle from each mobile node).

Mobispace location management does not consider continous position data, only the binary relation "inside/not inside"

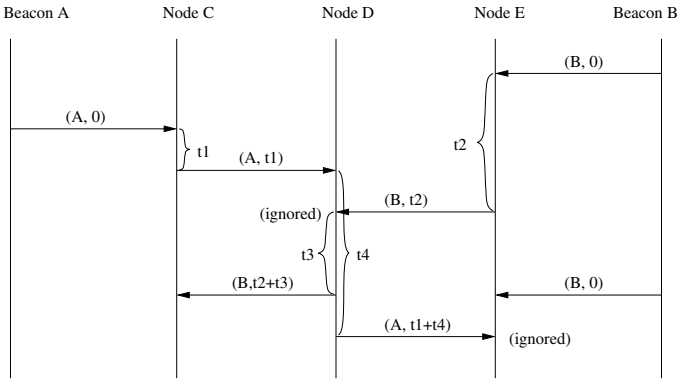

Fig. 2. Diffusion of localization information

a zone. The arithmetic calculation of positions may be therefore be replaced by a $(zone, age)$ pair. The age value is the (accumulated) time since the beacon of this zone was heard and serves as an estimated accuracy of the zone designation; a higher value is less trustworthy since the mobile node may have traveled far in the meantime.

During a secondary replication session, two mobile nodes C and D exchange their $(zone, age)$ pairs. Node C will accept D's zone data if

$$\begin{aligned} C_{age} - D_{age} &> Age\_margin \\ D_{age} &< Age\_maximum \end{aligned}$$

The Age_margin is a positive value which introduces a hysteresis of the system in order to avoid false zone changes. The Age_maximum restricts the age of the zone data so that outdated information is not diffused. Figure 2 shows a sequence diagram over the diffusion process. The arrows show the transport of zone-info between the nodes and the arrow label shows the value of the $(zone, age)$ pair. The two beacons A and B are within radio range of the two nodes C and E, while node D does not discover any beacons and is initially without a zone association.

The figure shows a scenario where nodes C and E discover the two beacons and associate themselves with the respective zones, and later during secondary replication offer their location data to node D. Of the two offers, node D accepts data from C and ignores that from E, since the data from C is more recent $(t_1 < t_2)$.

Node D will at a later instant try to inform its neighbours about its zone, which does not make any difference for node C. E will also ignore this information, since it has more recent data from beacon B.

A. Stability of diffused data

From Figure 2 it is apparent that if the two last messages to node E were received in the opposite order, node E would for a while be tricked into believing that it belongs to zone A.

Analysis of the diffusion algorithm reveals situations where mobile nodes can be associated with incorrect (not the nearest) zone, but these are likely to be intermittent situation and the network of mobile nodes tend to stabilize on correct data. An important feature of the algorithm is the use of an Age_maximum value to filter out zone data from distant zones. The Age_maximum should be set to a small number of minutes, shorter than e.g. the duration of a car trip.

No. of nodes	Avg. time
1	5.1 sec
2	5.8 sec
3	5.9 sec

TABLE I
RESULTS FROM RESPONSIVENESS EXPERIMENT

No. of nodes	Avg. replication time
2	2.1 sec
3	4.2 sec
4	6.4 sec
5	9.5 sec
6	19.8 sec
7	31.3 sec
8	44.6 sec

TABLE II
RESULTS FROM SCALABILITY EXPERIMENT

VII. EXPERIMENTAL RESULTS

The text will now continue with a presentation of some experimental results from the system under discussion.

A. Responsiveness

The first experiment measured the required time for a secondary node to establish a connection with a beacon, receive the zone designation, remove the old user centric tuple and insert a new one. The time measured did not include secondary replication. The time was measured with a configuration consisting of a beacon node and 1-3 secondary nodes. The observed times are shown in table I.

B. Scalability

In order to measure the scalablity of Bluetooth-based replication, the Impronto Bluetooth simulator was used. Replication involving a different number of nodes were simulated and the result shown in table II. Since the number of replication sessions grows with the square number of nodes and they share the radio bandwidth, it should come as no surprise that the observed times indicate a $O(n^2)$ growth rate.

C. Effect of diffusion

In order to examine the correctness of the diffusion strategy (Section VI) with respect to positioning, a simple test environment was organized. A square (75x75 m) was measured and marked on a football field (Figure 3). Two laptop computers (beacon A and B) were located in opposite corners and the diagonal line between them separated the zones, respectively zone A and B. Two persons equipped with mobile devices conducted the experiment. A Mobispace instance and a simple client application showing the current zone was implemented

No. of observations	Regular mov.	Irregular mov.
Both nodes showed correct zone	93	47
One node showed correct zone	7	48
No node showed correct zone	0	5

TABLE III
RESULTS FROM DIFFUSION EXPERIMENT

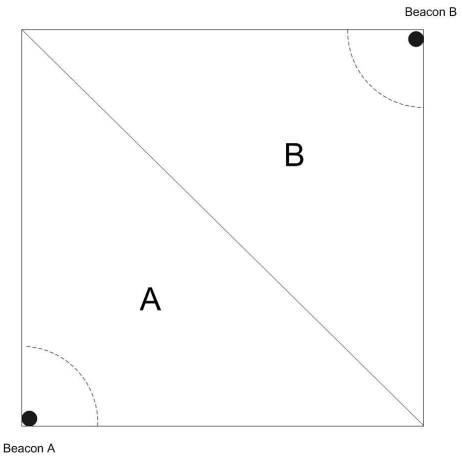

Fig. 3. A 75x75 m field with beacons in opposite corners

on the devices. The results obtained from this experiment are shown in table III.

Person p1 and p2 started walking from different corners in respectively zone A and B. p1 was given a head start by 10 seconds. Both moved in slow walking speed, in a straight line, towards the opposite beacon. When the devices discovered the beacon and the new zone was registered by the application, they turned and walked back to the starting point where the same actions were repeated. Each time p1 and p2 met and their devices exchanged (zone, age) pairs, they checked if the zone registered by the application corresponded with the correct side of the diagonal line. These observations were noted with either true or false. True indicated that the application showed the same zone as the person actually was located in, while false indicated that the zone registered by the application was wrong. The described actions were performed until they had 100 observation of true or false each.

A second set of observations was obtained when the movement patterns of the two persons was irregular. P1 and p2 consciously made change of direction and variations in movement speed. The table shows the result of this second set of observations.

VIII. CONCLUSION

This paper has discussed the Mobispace middleware in the context of location-aware distributed applications. The research effort has exploited a middleware for distributed tuplespace and the short range of Bluetooth tranceivers for location management purposes.

Although Bluetooth discovery mechanisms are not ideally suited for applications that require fast discovery of a large number of mobile units, the deployment scale of Bluetooth-equipped units make them interesting alternatives for location-aware mobile applications.

The research presented in this paper offers an integrated approach to tuplespace-based distributed systems and location management, in the sense that the the tuplespace offers the client location-sensitive tuples which are transparently representing properties associated with the current location of the mobile client. It also offers eay access to other location management information like who is in a given location (zone) and where a particular node is located.

The experiments that have been presented have validated the design and given indications on the scalability and reliability of the system.

REFERENCES

[1] A. Fongen and S. Taylor, "A distributed tuplespace for j2me environments," in *16th IASTED International Conference on Parallel and Distributed Computing and Systems*, Phoenix, AZ, 2005.
[2] D. Gelernter, "Generative communication in linda," *ACM Trans. Program. Lang. Syst.*, vol. 7, no. 1, pp. 80–112, 1985.
[3] P. Bishop and N. Warren, *JavaSpaces in practice*. Addison Wesley Longman Inc., 2003.
[4] S. Microsystems, "Javaspaces," available from: http://www.sun.com/software/jini/specs/jini1.2html/js-title.html [Jun 21, 2005].
[5] E. Freeman, S. Hupfer, and K. Arnold, *JavaSpaces Principles, Patterns and Practice*. Essex, UK, UK: Addison Wesley Longman Inc., 1999.
[6] IBM, "Tspaces," available from: http://www.almaden.ibm.com/cs/TSpaces/ [Jun 20, 2005].
[7] L. I. Patterson, R. S. Turner, and R. M. Hyatt, "Construction of a fault-tolerant distributed tuple-space," in *SAC '93: Proceedings of the 1993 ACM/SIGAPP symposium on Applied computing*. New York, NY, USA: ACM Press, 1993, pp. 279–285.
[8] G. P. Picco, A. L. Murphy, and G.-C. Roman, "Lime: Linda meets mobility," in *ICSE '99: Proceedings of the 21st international conference on Software engineering*. Los Alamitos, CA, USA: IEEE Computer Society Press, 1999, pp. 368–377.
[9] A. Charles, R. Menezes, and R. Tolksdorf, "On the implementation of swarmlinda," in *ACM Southeastern Conference (ACM-SE)*, Huntsville, AL, 2004.
[10] A. Omicini, "On the semantics of tuple-based coordination models," in *SAC '99: Proceedings of the 1999 ACM symposium on Applied computing*. New York, NY, USA: ACM Press, 1999, pp. 175–182.
[11] A. Göker, S. Watt, H. I. Myrhaug, N. Whitehead, M. Yakici, R. Bierig, S. K. Nuti, and H. Cumming, "An ambient, personalised, and context-sensitive information system for mobile users," in *EUSAI '04: Proceedings of the 2nd European Union symposium on Ambient intelligence*. New York, NY, USA: ACM Press, 2004, pp. 19–24.
[12] R. Bruno and F. Delmastro, "Design and analysis of a bluetooth-based indoor localization system." in *Personal Wireless Communications, IFIP-TC6 8th International Conference, PWC 2003, Venice, Italy, September 23-25, 2003, Proceedings*, 2003, pp. 711–725.
[13] M. Nilsson, J. Hallberg, and K. Synnes, "Bluetooth positioning," in *CSEE 2002*, 2002.
[14] L. Aalto, N. Göthlin, J. Korhonen, and T. Ojala, "Bluetooth and wap push based location-aware mobile advertising system," in *MobiSys '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*. New York, NY, USA: ACM Press, 2004, pp. 49–58.
[15] A. T. S. Chan, H. V. Leong, J. Chan, A. Hon, L. Lau, and L. Li, "Bluepoint: a bluetooth-based architecture for location-positioning services," in *SAC '03: Proceedings of the 2003 ACM symposium on Applied computing*. New York, NY, USA: ACM Press, 2003, pp. 990–995.
[16] M. Spratt, "An overview of positioning by diffusion," *Wirel. Netw.*, vol. 9, no. 6, pp. 565–574, 2003.