Context-Based Spontaneous Services in Ubiquitous Environments: Some Application Examples

Tuan Dung Nguyen
Institut TELECOM ; TELECOM Bretagne
Technopole de Brest Iroise, CS83818, 29238 Brest, France
Email: td.nguyen@telecom-bretagne.eu

Siegfried Rouvrais
Institut TELECOM ; TELECOM Bretagne
Technopole de Brest Iroise, CS83818, 29238 Brest, France
Email: siegfried.rouvrais@telecom-bretagne.eu

***Abstract*—Today, people often go around with different personal portable or wearable computing devices. They can interact wirelessly with static and mobile sensors gradually integrated in the surrounding environments that provide them with various context information characterizing their current situation. Therefore, context-based spontaneous services help applications provide users with the most relevant information without user's specific intervention. This paper illustrates the interest of such services in mobile and ubiquitous environments through several examples in which nearby heterogeneous mobile devices and sensors communicate in an autonomous fashion. Their proof-of-concept prototypes have shown the benefits of our previously proposed middleware for context management.**

I. Introduction

Wireless communication advances enables the creation of ad hoc networks operating independently from any preestablished infrastructures. Such environments are created by a group of people collaborating to provide themselves with some spontaneous services (e.g., resource sharing, collaborative workspace) [1]. Moreover, the Weiser's vision of ubiquitous computing [2] is gradually coming closer to reality as people are now surrounded by more and more computing-capable objects. The latter devices range from different portable/wearable computing devices (e.g., PDAs, mobile music player) to tiny sensors seamlessly integrated in surrounding environments (e.g., RFID tags, motes).

Wireless sensor networks were initially motivated by military applications, e.g., battlefield surveillance, enemy tracking. However, significant technology advances have resulted in the increasing use of small-size and inexpensive sensors in civil application areas (e.g., environmental monitoring, industrial supply chain management). This trend has been further propelled recently with research on people-centric sensing applications [3], [4], [5]. The use of sensors (e.g., RFID tags, motes) is no longer targeted to only small-scale and specialized tasks but also to large-scale daily-life applications. Such applications exploit various static and mobile sensors to discover raw context data characterizing user's current situation (e.g., location, temperature). Such low-level context information is then transferred to a centralized entity for processing to obtain high-level context information used in some specific context-based services (e.g., traffic, pollution surveillance).

The ubiquitous sensing capacity promotes at the same time context-based spontaneous services functioning on each user's device using available short to medium-range network connections. These services are capable of exploiting various context information to provide "task-relevant information and/or services to a user" [6] in a decentralized fashion. For example, a personal heath-care service can exploit information about user's health conditions (e.g., heart rate, temperature) and local weather condition (e.g., ambient temperature, heat index) to smartly adapt the scheduled activities. In intelligent transport systems, a driving assistant can use up-to-date information about the weather condition, road status, speed from nearby vehicles to propose the most relevant recommendations (e.g., route finding, traffic jam alert). Unlike the centralized approach, context-based spontaneous services alleviate some privacy concerns as sensitive context information is not managed by a centralized authority (i.e., Big Brother issue). Furthermore, users are motivated to participate in the context information sensing and processing due to the benefit of their own spontaneous context-based services.

This paper presents some examples of context-based spontaneous services and discusses their challenges in such environments. Then, it motivates the use of a middleware to provide common context management services and underlines its benefit through the prototype development of the aforementioned services.

The rest of this paper is organized as follows. First, Section II presents examples of context-aware spontaneous services as well as the required context information. Next, Section III motivates and describes the use of our previously proposed middleware for context management. Then, Section IV describes and evaluates the prototype of our presented services. We present some related works in Section VI before finally concluding the paper in the last Section.

II. Context-based spontaneous services

A. Spontaneous messenger

Figure 1 illustrates a scenario where Bob's device and Alice's form an infrastructureless ad hoc network. John is at that moment reachable from Bob via Alice in a multihop connection. John and Peter attend the same university course so they often see each other in the class. Both Peter and Alice are in the buddy list of Bob. Our messenger service

aims at transmitting online/offline messages between users in a decentralized manner while taking into account two context information: *security level* and *frequent contact*. An offline message to Peter will be transmitted to his frequent contacts (1) as well as his frequently visited places (e.g., classroom, library) (2). We suppose that a message can only be sent if the message's security level is lower than the message's destination's (cf. Bell LaPadula model).

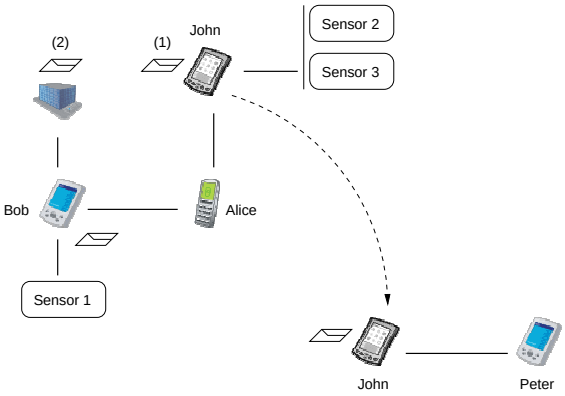


Fig. 1. Application scenario

B. Elderly assistant

This service is capable of monitoring the following context information: *air temperature*, *relative humidity* and *heat index*. This service can trigger an alert according to the aforementioned impact levels to the health or raise that information to other services in order to adapt their functions to the situation. As depicted in Figure 1, some context information (e.g., relative humidity) may be not directly available from the corresponding sensor due to user's device limited hardware capacity (e.g., lack of required networking capacity) or being out-of-range.

C. Shopping assistant

This service exploits *shopping discount* context information to provide the user with appropriate shopping hints when she/he accesses to the shopping list. The related information is captured by each individual on the move and collaboratively shared among the interested entities.

D. Categories of context information

Context can be defined as "any information that can be used to characterize the situation of an entity, where an entity can be a person, place, physical or computational object" [6]. Primitive context information can be retrieved from hardware sensors, e.g., user's geographical location from a GPS receiver. It can also be retrieved from software sensors, e.g., device's processor usage obtained from the operating system. Composite context information is higher-level information that must be obtained from the composition of different composite or primitive context information (e.g., the heat index is calculated from the air temperature and the relative humidity [7]).

1) User-related:

- *User availability*. This primitive context information is provided by the user to inform others about his/her current status (e.g., available, idle, busy).
- *Location*. This primitive context information depicts the user's current location, i.e., in a meeting room, in a supermarket. It is provided by hardware sensors such as GPS receivers or RFID beacons.
- *Frequent contact*. This context information shows the user's most frequent contact list according to his/her social relations, i.e., classmate, colleague. It is provided by a software sensor monitoring user's physical neighbors.

2) Execution-related:

- *Battery level*. This context information shows the device's available battery in percentage (%). In our application, this one is used to adapt the application's function as well as the underlying middleware's.
- *Security level*. This context information shows the device's security level which ranges from 1 to 7. This one is inherently provided by the system.

3) Environment-related:

- *Air temperature*. This context information shows the current air temperature in Celsius degree (°C). It is provided by a temperature sensor.
- *Relative humidity*. This context information shows the current relative humidity in percentage (%). It is provided by a humidity sensor.
- *Heat index*. This composite context information ($HI = f(T, R)$) is calculated from air temperature and relative humidity to determine the human-perceived temperature [7].
  The risk categories of heat index to human health are then defined as follows:
  - from 80 to 90°F (27 – 32°C): Caution.
  - from 90 to 105°F (32 – 41°C): Extreme Caution.
  - from 105 to 130°F (41 – 54°C): Danger.
  - 130°F or higher (54°C or higher): Extreme Danger.
- *Shopping discount*. This context information shows current shopping discount hints provided either by a special hardware sensor located in each shopping place or by the user's annotation.

III. Middleware for context management

Managing context information in such environments must cope with issues related to the sensor accessibility and availability. First, due to the heterogeneity of devices and connections, a mobile device may not be capable of discovering and gaining access to a required sensor because it is not locally integrated and demands an unsupported network connection, e.g., ZigBee, EnOcean (i.e., sensor accesibility). Second, a sensor may become unavailable to a mobile device because of a sensor's failure or they are out-of-range (i.e., sensor availability). A device's local context information should be shared with other devices in order to obtain a more complete

context, e.g., frequent contact, shopping discount. Furthermore, it is not always possible to carry out locally context information processing operations due to the device's limited physical capacity or available battery. For example, it is of interest to obtain the heat index from other devices because the composition function f is not supported on several low-end mobile devices.

The general architecture is depicted in Figure 2 where different context-based software services, e.g., spontaneous messenger, elderly assistant and shopping assistant already presented in this paper, are developed and executed on top of the two following middleware services: a peer-to-peer communication and a context management system.

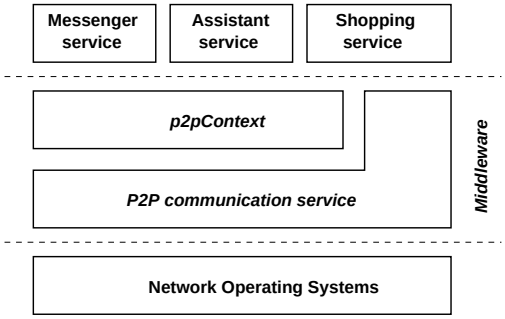


Fig. 2. Software application architecture

In previous work, we have proposed *p2pContext*, a component middleware for peer-to-peer context management in mobile and ubiquitous environments [8]. Its component-based architecture and peer-to-peer dissemination of context information using multiple overlays facilitate the construction of different context-aware applications (cf. Figure 3). Context information requirements are systematically managed independently from the sensor accessibility and availability. We distinguish two principal roles: application developers and sensor developers. At design time, application developers specify their required context peers independently from the aforementioned sensor accessibility and availability issues. Sensor developers are responsible for sensor capacity implementation, e.g., how to connect to a local GPS receiver, how to discover and gain access to a remote wireless sensor.

Figure 4 depicts the middleware architecture with two principal layers: context management and sensor access. The interaction detail with a specific sensor is encapsulated in a software component (e.g., sensor discovery, sensor access). Applications provide their context information requirement through a specification in XML format (cf. Figure 5). The middleware provides the required information (i.e., primitive or composite context) by automatically managing (creation, binding, removal) different software components according to the device's capacity and context requirements. Context information is shared among devices of common interest with different levels of confidence (e.g., directly from sensors, indirectly from other devices). Context-based services can obtain context information in *push* or *pull* mode. In the former,

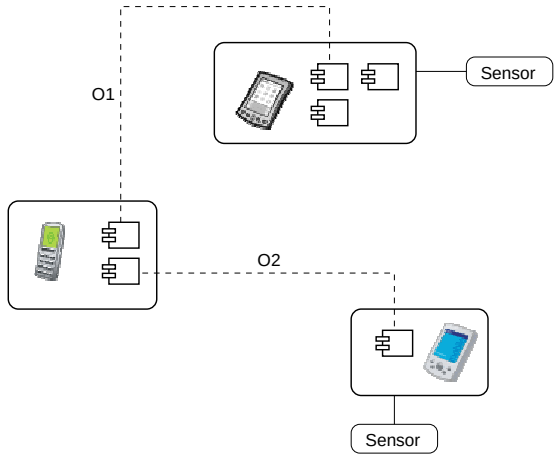


Fig. 3. p2pContext context management

context information is automatically delivered to them whenever its value changes. In the latter, context-based services get context information on demand through an API call provided by the middleware. This paper does not focus on the technical details about this middleware service but on its benefits in building various context-based spontaneous services.

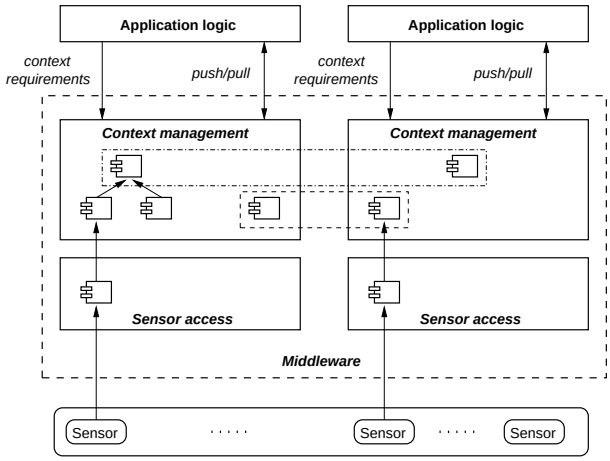


Fig. 4. Middleware architecture

A. Peer-to-peer communication

JXTA (Juxtapose) [9] is a peer-to-peer platform initially designed by Sun as a set of communication protocols independent from operating systems and programming languages. Several implementations now exist in different languages (e.g. Java, C, Perl) and are adapted to different hardware platforms (e.g. computers, mobiles or PDAs). JXME (JXTA for J2ME) [10] is the adapted version of JXTA for mobile devices. Its current version supports two ways of communication: proxy-based or proxyless (cf. Figure 6). The proxy-based version supports various telephone mobiles with CLDC profile but every communication between devices must pass through

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sensor SYSTEM "sensor.dtd">
<sensorAccessibility>
 <sensor name="temperature">
  <type>IntegerCxtData</type>
  <implementation>
   org.telb.demo.sensor.TemperatureSensor
  </implementation>
 </sensor>
</sensorAccessibility>

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE context SYSTEM "context.dtd">
<contextRequirement>
 <context name="temperature">
  <category>primitive</category>
  <privacy>shared</privacy>
  <ttl>600</ttl>
  <scope>3</scope>
  <type>IntegerCxtData</type>
 </context>

 <context name="heatIndex">
  <category>composite</category>
  <privacy>shared</privacy>
  <ttl>600</ttl>
  <scope>3</scope>
  <type>IntegerCxtData</type>
  <composition>
   <operand>temperature</operand>
   <operand>relativeHumidity</operand>
   <implementation>
    org.telb.demo.operator.HeatIndexComposition
   </implementation>
  </composition>
 </context>
</contextRequirement>
```

Fig. 5. Specification examples

a proxy. The proxyless version enables direct communication but supports only PDAs with a CDC profile. In our prototype, we use the proxyless version in order to better support the decentralized characteristic of spontaneous networks.

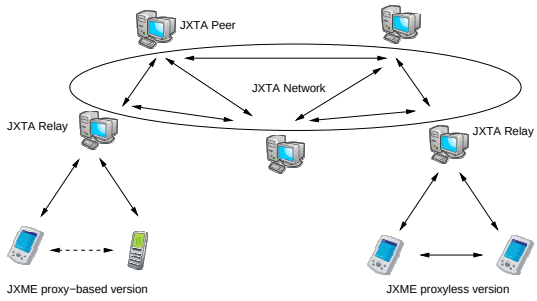


Fig. 6. JXTA and JXME network

IV. Prototype and Evaluation

We implemented a prototype version in Java using a Julia, a reference implementation of Fractal component framework [11]. Then the middleware was used to develop the spontaneous software services presented in Section II. The whole system is executed on an experimental testbed consists of PDA devices (HP iPAQ hx2700 with 624 MHz processor, 64 MB RAM, Windows Mobile 5 and IBM J9 Java virtual machine). Those mobile devices are connected in IEEE 802.11b wireless ad hoc networks.

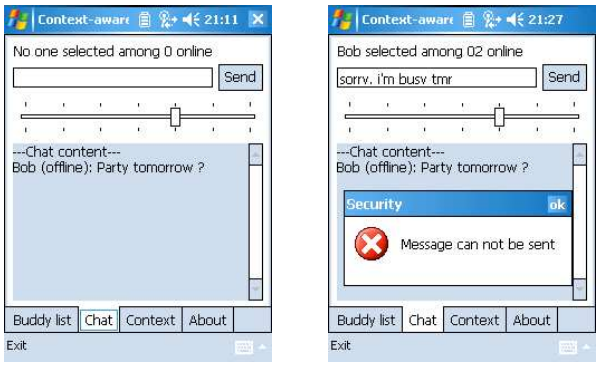
The middleware facilitates the development by allowing the specification of sensor accessibility and context requirements. It is responsible for initializing and binding necessary software components to provide the required context information.

The messenger's user interface is depicted in Figure 7 where users can see their online buddy list maintained dynamically in a peer-to-peer manner.

(a) (b)

Fig. 7. Main user interface

Figure 8 illustrates the messaging function where users can send a message to an available contact in their buddy list. Each message is sent with a security level (from 1 to 7). A warning message will be shown if the message cannot be sent due to the security policy. Moreover, users can also send an offline message to an unreachable contact. The messenger service sends it to physical neighbor nodes which are frequent contacts of the destination. The latter will cache the message during a timeout period in order to retransmit it to the corresponding destination.

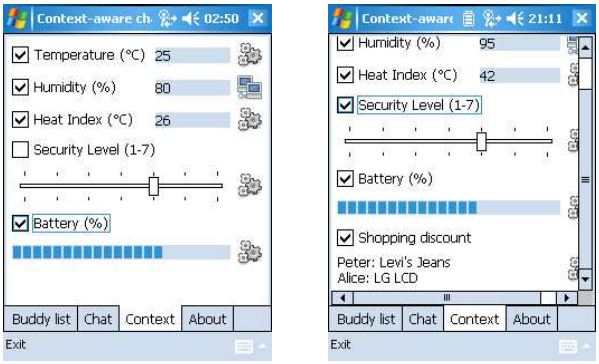


(a) (b)

Fig. 8. Messenger interface

Figure 9 depicts the context information display. On the left hand side, user can choose the required context information using the respective check box. On the right hand side of each context information, a small icon illustrates the corresponding sensor capacity. For instance, the mobile device has access

to a temperature sensor but does not gain access to a humidity sensor. However, the middleware manage to provide the required context information from another peer in the corresponding overlay. The context management middleware allows user to add new context information requirement as well as new sensors without interrupting currently running services, e.g., shopping discount.



(a) (b)

Fig. 9. Context information interface

V. Discussion

A. Application development experiences

We distinguish two principal roles: application developers and sensor developers. At design time, application developers specify their required context peers without caring about the sensor capacities of heterogeneous mobile devices. Sensor developers are responsible for sensor capacity implementation, e.g., how to connect to a local GPS receiver, how to discover and access to a remote wireless sensor. All of them have access to a context base of context concepts and implementations. This avoid any ambiguity between different concepts and facilitate the reuse. Application developers can focus on the business logic as the context information is systematically managed by the middleware. It enables the deployment on heterogeneous devices according to the device's physical capacities and context requirement. Later, context information is collaboratively acquired and shared among interested entities.

B. Scalability and performance

Context information is selectively shared in different peer-to-peer overlays. This mechanism helps restrict the communication overhead to only interested entities. The middleware prototype presented in this paper based on JXME to manage the different context overlays. Our first performance analysis has shown the limits of this approach which incurs redundant communication overhead. The reason is that wireless spontaneous networks have distinct characteristics making existing algorithms, e.g., JXME, not really relevant. For instance, an end-to-end connectivity between two given nodes is not always guaranteed. Moreover, multihop routing protocols incur costly communication, especially in networks with frequent mobility [12]. Therefore, a new mechanism was recently proposed for efficient context overlay management in these environments.

VI. Related work

Context-awareness has attracted a lot of research work in distributed systems. However, to the best of our knowledge, none of them has fully taken into account the sensor accessibility and sensor availability issues for context management in mobile and ubiquitous environments.

Context Toolkit [13] provides the first context abstraction to facilitate the development and deployment of context-aware applications. The authors proposed context widgets to encapsulate the sensing details and allow other components to retrieve context information. However, a context widget must be statically connected to a sensing facility. In our proposal, we provide two abstraction levels to enable the sharing of context information among distributed nodes.

NORS [14] is an open source platform to facilitate participatory sensing with mobile phones. However, NORS transmits the collected data to specific application servers instead of peer-to-peer decentralized data sharing among mobile devices as in our work. We also support composite context information and heterogeneous mobile devices.

Urbanet [15] exploits sensors in a decentralized fashion as in our work. However, the authors proposed to integrate all the sensing strategies or migrate the service from a device to another while we propose to process and share various context information (e.g., primitive, composite) in separate peer-to-peer overlays.

ConChat [16] motivates the exploitation of context-awareness to improve the functions of a chat application through a set of context information (e.g. location, timezone, temperature, social activity). However, it does not support spontaneous networks and more complex context information as in our work.

BlueIRC [17] is also a peer-to-peer chat application built on top of JXME but it uses Bluetooth for communication. The main objective is to overcome the limitations of Bluetooth in terms of number of interconnectable devices and transmission range using the functions provided by JXME. Unlike our work, this application uses the proxy-based version of JXME and does not take into account the context-awareness issue.

VII. Conclusion

Context-based services are one of important means to provide relevant information to the user in mobile and ubiquitous computing. However, the heterogeneity and mobility of devices have raised also new challenges for context management. We have proposed in this paper several context-based spontaneous services that exploits context information coming from heterogeneous sensors in the user's surrounding environment to provide relevant service to user's situation. The preliminary evaluation has proved the feasibility and the benefits of our supported middleware to facilitate the application's construction in such dynamic environments. Future work consists of making further performance test (e.g., by changing the number of overlays) to fully validate our approach.

REFERENCES

[1] L. M. Feeney, B. Ahlgren, and A. Westerlund, "Spontaneous Networking: An Application-Oriented Approach to Ad Hoc Networking," *IEEE Communications Magazine*, vol. 39, no. 6, pp. 176–181, Jun. 2001.

[2] M. Weiser, "Some Computer Science Problems in Ubiquitous Computing," *Communications of the ACM*, vol. 36, no. 7, pp. 74–84, Jul. 1993.

[3] Microsoft Research, "SenseWeb Project," Web site, consulted in 2008, http://research.microsoft.com/nec/senseWeb.

[4] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden, "CarTel: A Distributed Mobile Sensor Computing System," in *Proc. 4th ACM Int. Conf. on Embedded Networked Sensor Systems (SenSys'06)*. Boulder, CO, USA: ACM Press, Oct. 2006.

[5] C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin, and N. Triandopoulos, "AnonySense: Privacy-Aware People-Centric Sensing," in *Proc. 6th ACM/USENIX Int. Conf. on Mobile Systems, Applications, and Services (MobiSys'08)*. Breckenridge, CO, USA: ACM Press, Jun. 2008.

[6] G. D. Abowd and A. K. Dey, "Towards a Better Understanding of Context and Context-Awareness," in *Proc. 1st Int. Symp. on Handheld and Ubiquitous Computing (HUC'99)*, ser. LNCS, vol. 1707. Karlsruhe, Germany: Springer Verlag, Sep. 1999, pp. 304–307.

[7] NOAA's National Weather Service, USA, Web site, consulted in 2008, http://www.crh.noaa.gov.

[8] T. D. Nguyen and S. Rouvrais, "Towards a Peer-to-peer Middleware for Context Provisioning in Spontaneous Networks," in *Proc. 5th Workshop on Middleware for Network Eccentric and Mobile Applications (ESF/MiNEMA'07)*, Magdeburg, Germany, Sep. 2007, pp. 54–57.

[9] JXTA, Web site, consulted in 2008, http://www.jxta.org.

[10] JXME, Web site, consulted in 2008, https://jxta-jxme.dev.java.net.

[11] Fractal. Fractal Component Model, Web site, consulted in 2008, http://fractal.objectweb.org.

[12] R. Friedman, D. Gavidia, L. Rodrigues, A. C. Viana, and S. Voulgaris, "Gossiping on MANETs: the Beauty and the Beast," *ACM SIGOPS Operating Systems Review*, vol. 41, no. 5, pp. 67–74, 2007.

[13] A. K. Dey, D. Salber, and G. D. Abowd, "A Conceptual Framework and Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications," *Human-Computer Interaction*, vol. 16, no. 2–4, pp. 97–166, 2001.

[14] D. Trossen and D. Pavel, "NORS: An Open Source Platform to Facilitate Participatory Sensing with Mobile Phones," in *Proc. 4th Int. Conf. on Mobile and Ubiquitous Systems: Computing, Networking and Services (Mobiquitous'07)*. Philadelphia, PA, USA: IEEE CS Press, Aug. 2007, pp. 1–8.

[15] O. Riva and C. Borcea, "The Urbanet Revolution: Sensor Power to the People," *IEEE Pervasive Computing*, vol. 6, no. 2, pp. 41–49, 2007.

[16] A. Ranganathan, R. H. Campbell, A. Ravi, and A. Mahajan, "ConChat: A Context-Aware Chat Program," *IEEE Pervasive Computing*, vol. 1, no. 3, pp. 51–57, Jul. 2002.

[17] C. Blundo and E. D. Cristofaro, "A Bluetooth-Based JXME Infrastructure," in *Proc. 9th OTM Int. Symp. on Distributed Objects and Applications (DOA'07)*, ser. LNCS, vol. 4803. Vilamoura, Portugal: Springer Verlag, Nov. 2007, pp. 667–682.