Mobile Gaia: A Middleware for Ad-hoc Pervasive Computing

Shiva Chetan, Jalal Al-Muhtadi, Roy Campbell, M. Dennis Mickunas
University of Illinois at Urbana-Champaign
{chetan, almuhtad, rhc, mickunas}@cs.uiuc.edu

Abstract—**Pervasive Computing promotes an environment that blurs the distinction between digital and physical devices and integrates all entities in a physical space into a cohesive programmable unit. Some of the early research activities in pervasive computing focused on developing infrastructures for pervasive applications. These infrastructures successfully merged physical and digital entities in an environment to create aware homes, smart offices and active spaces. In recent years, ad-hoc pervasive computing has attracted attention with the proliferation of low cost, short-range wireless devices. Ad-hoc pervasive computing does not assume digital devices to be tied to physical environments and aims to create digital "clusters" that can be viewed as a unified entity. The user can program this cluster of devices with a single programming interface. In this paper, we introduce our middleware, called Mobile Gaia, for ad-hoc pervasive computing. Mobile Gaia is a services-based middleware that integrates resources of various devices. It manages several functions such as forming and maintaining device collections, sharing resources among devices and enables seamless service interactions. It also provides an application framework to develop applications for the device collection. The application framework decomposes the application into smaller components that can run on different devices in this collection. We discuss the architecture of Mobile Gaia and introduce a sample application that has been designed using our middleware.**

Ubiquitous computing; personal active spaces, ad-hoc networking; Gaia; mobility; middleware.

I. Introduction

The traditional notion of pervasive computing is a digitally-enhanced habitat where physical and digital devices are seamlessly integrated. Physical devices such as lights, doors, electrical appliances and sensors are integrated with digital devices such as handhelds, mobile phones and laptop computers. Physical devices can be controlled by their digital counterparts and digital devices acquire information about the environment from physical devices. This vision has been realized in various projects such as Active Spaces [1], Aware Home [2] and Interactive Workspaces [3].

In recent years, there has been a surge of interest in ad-hoc pervasive systems [4, 5]. These systems do not assume a particular infrastructure and devices can be grouped together without an underlying infrastructure. These systems are well suited to scenarios where devices are grouped based on particular criteria. Examples of such systems include cluster of devices belonging to a single user, devices grouped for a particular application such as pervasive gaming and data sharing and device ensembles for resource sharing.

In this paper, we present our middleware, called Mobile Gaia, for ad-hoc pervasive computing. Our notion of ad-hoc pervasive computing is a cluster of personal devices that can communicate and share resources among each other. The user can program the cluster through a common interface. We refer to this cluster as a personal active space. Mobile Gaia contains services to discover devices that form the personal space, maintain the composition of the cluster, share resources among devices in the cluster and facilitate communication.

Each personal active space has a coordinator device and zero or more client devices. The coordinator maintains the composition of the cluster. It expects periodic heartbeat messages from all client devices, discovers new devices in the vicinity and enables resources to be shared among devices in the cluster. Any device that has the required resources can act as the coordinator of the cluster.

Services in Mobile Gaia are decomposed into components. This allows only required components of services to be loaded into the middleware thus reducing memory and battery power requirements. Component-based middleware has been found suitable for mobile devices from our past research [6, 7]. Componentization is achieved by identifying independent units of a service whose composition results in the original service. For example, location service is a composition of sensor fusion, spatial model and location adaptor components.

Services in Mobile Gaia are present in two roles – coordinator and client. When a device is a coordinator of a cluster, the services on the device are present in coordinator role while they are present in the client role when the device is a client in a cluster. In coordinator role, services have additional responsibilities such as integrating similar services on all devices in the cluster, managing services in all devices and so on. For example, location service of a coordinator fuses sensor information from multiple location sensors in the cluster and maintains a spatial model in addition to receiving location information from various location sensors. Similarly, event service of a coordinator device maintains a database of all event channels in the device cluster. These functionalities are not present in services of client devices. Location service of a client device just sends information from its location sensors to the coordinator device. Therefore, the location service of the client device contains only the location adaptor component that reads location data from its sensor.

We have found our component-based service architecture suitable for the above requirements. When a device is the coordinator of a cluster, required service components are dynamically loaded into the middleware. Similarly, coordinator components are unloaded and client components are loaded if the device becomes a client in the cluster. We have designed a service deployment framework that manages the services of Mobile Gaia. The framework keeps track of all service components loaded in the middleware. When the role of a device changes from coordinator to client or vice versa, the cluster management service notifies the framework of this change. The framework unloads the current service components and loads the new set of components into the middleware.

There are a few pervasive computing products that facilitate interactions among devices over ad-hoc networks. These include Bluetooth and IrDA-based devices, mobile phones with GPRS and so on. While these products enable interactions among devices and allow formation of communicating clusters they do not provide a uniform abstraction to similar resources in the cluster. Our notion of a personal active space provides a programming and usage abstraction to all devices in the cluster. Applications and services can use resources of the devices in the cluster through a common abstraction. This enables several functionalities such as sharing location information among different devices in the personal active space, authenticating all devices in the space using a single authentication process and so on.

Section 2 discusses the architecture of Mobile Gaia. Section 3 discusses how services interact. Section 4 explains the personal active space management protocols. Section 5 describes Mobile Gaia application framework. Section 6 discusses a sample Mobile Gaia application. Section 7 outlines some related work. Section 8 concludes.

II. Mobile Gaia Architecture

Mobile Gaia consists of a set of core services that manages the device cluster. These services enable the devices in the cluster to share resources and data seamlessly. The architecture of Mobile Gaia is shown in Figure 1.

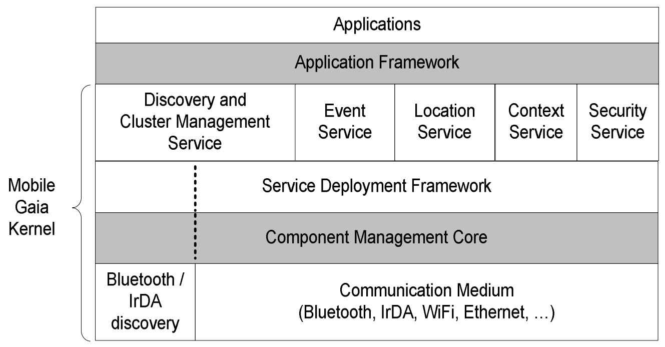
Applications
Application Framework
Discovery and Cluster Management Service
Event Service
Location Service
Context Service
Security Service
Service Deployment Framework
Component Management Core
Bluetooth / IrDA discovery
Communication Medium (Bluetooth, IrDA, WiFi, Ethernet, ...)
Mobile Gaia Kernel

Figure 1. Mobile Gaia Architecture

The core services make up the kernel of Mobile Gaia. The cluster management service discovers devices in its vicinity, negotiates with the devices to join the cluster and manages the composition of the cluster. It maintains a list of devices present in the cluster. Event service enables events to be communicated among devices in the cluster. Location and context services provide location and context information to all devices in the cluster. The security service provides authentication and access control services to the cluster. The service deployment framework acts as a container for these services. The component management core provides necessary low level support for creating and managing components.

Each device in the cluster contains a thin Mobile Gaia kernel that consists of service deployment framework, component management core and cluster management service to interact with the coordinator. A typical personal active space is shown in Figure 2, with a representative application.

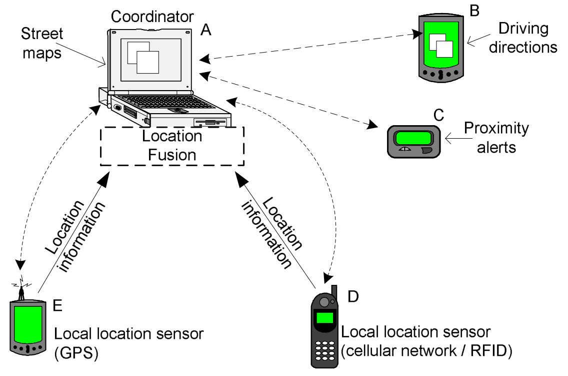

Figure 2. A typical personal active space. The laptop (A) is designated as a coordinator. All other devices are clients. Location information is captured by two different devices and utilized by a distributed driving directions application running in the personal space.

III. Service Interaction

Mobile Gaia supports seamless interaction among services in the device collection. Each service exists in two roles – coordinator role and client role. Mobile Gaia uses a service deployment framework that decides which role of the service to invoke based on whether the host device is a coordinator or a client of the cluster. Below we discuss the service deployment framework and how it facilitates interaction among various services of Mobile Gaia.

A. Service Deployment Framework

Services in Mobile Gaia are implemented as components. The service deployment framework forms a container for these service components. It manages installation of new service components, loading and unloading of components and removing components when they are no longer needed. The service deployment framework is based on the "What You Need Is What You Get" (WYNIWYG) model [6]. In the WYNIWYG model, a system loads a minimal set of components required to provide a certain service to applications. The WYNIWYG model was used in the 2K Operating System [8], which was a precursor to the Gaia Operating System [1]. The WYNIWYG model was found to be suitable for integrating resource-constrained devices such as PDAs, cell phones and other mobile devices into infrastructure-based distributed systems [7]. We are reusing the WYNIWYG model to integrate mobile devices into infrastructure-free distributed systems.

The Service Deployment Architecture is shown in Figure 3. Each service in Mobile Gaia consists of a service wrapper component and a set of service components. The service wrapper component provides interfaces to access the service components. Therefore, the service wrapper is dependent on the service components to export the intended services. For example, when a GPS-enabled laptop becomes the coordinator of a cluster, the location service in the laptop is responsible for fusing sensor information from different devices of the cluster along with the GPS information from the laptop. Therefore, the location service of the laptop consists of the sensor fusion, spatial model and GPS-adaptor components. If the laptop becomes a client device to a cluster, the sensor fusion and spatial model components are unloaded by the deployment framework and only the GPS-adaptor component is retained in the location service. The location service wrapper provides appropriate interfaces to the location service in the coordinator and client roles.

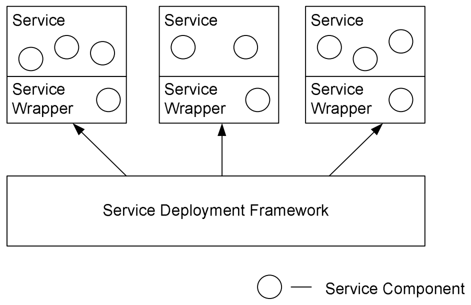

Figure 3. Service Deployment Architecture

When a service is deployed, the service specifies the components that are to be loaded in the coordinator and client roles. Whenever a device changes roles from coordinator to client or vice versa, the cluster management service notifies the service deployment framework of this change. For each service, the service deployment framework checks to see if components have to be loaded/unloaded and makes appropriate changes. A typical service component specification is shown in Figure 4. The service deployment framework also facilitates resource sharing.

In the following sections, we discuss a few services of Mobile Gaia. For each service, we describe their functionalities and components in coordinator and client roles.

B. *Cluster Management Service*

A key capability of our system is the ability to discover nearby personal devices dynamically, and bootstrap a personal active space on the discovered devices. When new personal devices are within range, they are added to the personal active space on the fly. Depending on the capabilities of the new device, our framework loads appropriate components and runs additional services that allow the system to utilize the device's capabilities. Likewise, once a device is out of range, it is automatically removed from the personal active space. The functionality of discovering devices and forming clusters are handled by the cluster management service.

The cluster management service consists of different components (Figure 4). The discovery component provides a mechanism for discovering nearby devices. The discovery component utilizes different discovery protocols, including Bluetooth device discovery, IrDA, or a combined discovery protocol like the one proposed by Woodings et al. [9]. The cluster management protocol implements a mechanism for bootstrapping a personal active space (if none exists), for facilitating the dynamic addition and removal of personal devices and for selecting a coordinator for the cluster. The distributed space repository maintains a database of active devices and services in the personal active space. It keeps this information up-to-date through a heartbeat mechanism, where devices send periodic notifications to indicate their presence. Finally, a local component repository runs on client devices to maintain a list of components running locally on the device.

```
Coordinator:
  ClusterManagementProtocol
  DiscoveryComponent
  DistributedSpaceRepository
  HeartbeatReceiver
Client:
  ClusterManagementProtocol
  LocalComponentRepository
  HeartbeatSender
```

Figure 4. Cluster Management Component Specification

C. *Event Service*

The event service is a publish-subscribe middleware that is used to send events among devices in the cluster. These events include updates to location information, heartbeat messages informing that the device is part of the cluster and location-based events. Events are implemented using event channels, which are objects that decouple the publishers from the subscribers. When an application wants to send events, it creates an event channel in the host device. The event service of the host device registers the event channel with the coordinator. The event service in the coordinator maintains a list of event channels in the cluster. When an application wants to subscribe to an event channel, it queries the local event service which in turn queries the event service of the coordinator for event channels. The event service of the coordinator acts like a repository that contains information about various event channels in the cluster.

```
Coordinator:
  EventChannelWrapper
  EventChannelRepository
  EventChannelManager
Client:
  EventChannelWrapper
  EventChannelManager
```

Figure 5. Event Service Component Specification

The Event Service Component Specification is shown in Figure 5. The event service in both coordinator and client roles contains event channel wrapper and event channel manager components. The event channel manager provides interfaces to create and destroy event channels on the local device. It also supports interfaces to publish and subscribe to local and remote event channels. The coordinator contains the event channel repository component that maintains a database of all event channels in the cluster and their descriptions. This database is updated when an event channel is created or destroyed in any of the devices in the cluster.

D. *Location Service*

The location service is responsible for enabling location-awareness in the device cluster. The location service fuses location information from different devices in the cluster and provides a probabilistic value of location for the entire cluster [10]. Different devices in the cluster have different location-sensing technologies such as GPS, WiFi, RFID, Ubisense [11] and so on. Our location service fuses location information from different devices and arrives at a probabilistic estimate of the location information. This location information is conveyed to all devices in the cluster. This cooperative approach to sharing location information enables devices to be location-aware in heterogeneous location-sensing environments. The location service also supports a spatial database that contains a spatial model of the physical world. The spatial database enables location information to be associated with spatial information. Spatial information is required for route planning, finding points-of-interest and for location-based triggers. Location-based triggers are events that are generated when a certain spatial condition is satisfied. These events include object entering a certain region, object in the vicinity of another object and so on. Currently, we are exploring usage of multi-resolution spatial information for location-based triggers. Multi-resolution spatial information is useful for devices where storage is at a premium. Devices can download maps at different resolutions based on memory capacity of the device, bandwidth, battery power and so on.

The location service component specification is shown in Figure 6. When a device runs as a coordinator of a cluster, the location service in the device contains sensor fusion and spatial model components. In addition, the location service in both coordinator and client devices contains location sensor components that get information from location sensors attached to the devices. This includes adaptor components for GPS, WiFi, Radio frequency sensors and so on.

```
Coordinator:
  LocationServiceWrapper
  SensorFusionModule
  SpatialDatabaseModule
  {LocalLocationSensorModules}
Client:
  LocationServiceWrapper
  {LocalLocationSensorModules}
```

Figure 6. Location Service Component Specification

E. *Security Service*

Security is essential in personal active spaces. The security service can be divided into two main parts – authentication and access control. Device and space authentication is needed to ensure that only authorized devices are allowed to connect to a user's personal active space, and a device can only connect to personal spaces approved by its owner. Access control identifies what information or resources a device can share with a specific space name. We deploy a simple scheme for authenticating devices through public key, which we describe in section 4. For access control, users are allowed to create simple security policies on the personal device, which specify what services may run on the device when connected to a certain space. Security has the following components in both coordinator and client devices: DigitalSigner, SignatureValidator, and PolicyEnforcer.

IV. Personal Active Space Management Protocols

In this section we describe how clusters, or personal active spaces, are setup and bootstrapped.

A. *Setting up a Personal Active Space*

A personal active space consists of all discoverable personal devices, with a security policy that allows them to participate in a specific personal active space, e.g. Bob can configure his devices to only join "Bob's personal space" but no other spaces. In each personal active space, a device is designated as the coordinator of the space. The coordinator is responsible for cluster and service management. Ideally, the device with the most CPU, memory, and power resources should be selected as the space coordinator. The setup phase consists of the following steps:

1. The user identifies a name for his personal active space. A key pair is generated for this space. Key pairs are also generated for all devices.

2. On all client personal devices, the user installs the Mobile Gaia thin kernel, consisting of the client services. In addition, each client is configured with a list of spaces that they are permitted to connect to. For each space, the public key of that space is stored in the device. We assume that public keys are transmitted and stored in the device by the user using an out-of-band mechanism, e.g., typing it in, or using IrDA with confirmation and/or a pass code.

3. For each device that the user wishes to designate as a coordinator, the user needs to install the Mobile Gaia middleware, configure the space names that it is permitted to coordinate, and the public/private key pair associated with these space names. Because a user may designate more than one device to be a coordinator (this is useful in scenarios where a user forgets one of his devices, so another device can act as coordinator), the user is also expected to assign a 'preference value' to each coordinator, which indicates which device the user prefers to be designated the coordinator in case more than one coordinator device exists.

B. *Bootstrapping a Personal Active Space*

Our aim is to provide an automated method for bootstrapping a personal active space once personal devices are near each other and are able to communicate. This automated bootstrapping and management of devices provides a plug and play solution that does not require users to reconfigure their devices each time a personal active space is bootstrapped. The bootstrapping process works as follows.

1. The discovery service running on a designated coordinator machine discovers nearby devices.

2. Upon discovering a client device, an invitation is sent to join the space, along with the space name.

3. If the security policy on the target device permits connection to this space, a random number is sent back to the coordinator for authentication purposes.

4. The coordinator returns a packet containing its communication address and the random number signed with the space's digital signature. If the space is configured so that only authenticated devices are allowed to join, the device will need to send back a token signed by its private key, to achieve mutual authentication.

5. If a coordinator discovers another coordinator device for the same space name, then negotiations take place to identify which coordinator prevails. If one of the coordinators is in the 'bootstrapping phase' while the other has already bootstrapped, then the latter will take precedence.

6. Otherwise, a net value is calculated based on the 'preference value' a user assigns to each designated coordinator, combined with other factors including remaining power level, processing capability, etc. and the coordinator with the higher value prevails. The other coordinator utilizes the event service to notify all of its client devices of a coordinator switch. It then acts as a regular client device.

C. Adding and Removing Devices

Once the cluster is bootstrapped, when new devices are discovered the coordinator invites them to join, and a protocol similar to what was described above takes place. Active client devices are monitored through a heartbeat mechanism and are processed by the cluster management service at the coordinator. If the heartbeat of a particular device stops, the device times out and is removed from the cluster. If a device leaves a cluster, it becomes an 'orphan' device. Orphan clients wait for invitations to join clusters, whereas orphan coordinators start the bootstrapping process in an attempt to form a new cluster.

V. APPLICATION FRAMEWORK

In Gaia, we identified six patterns [12] that are needed for typical ubiquitous computing applications. These are multi-device support, user-centrism, run-time adaptation, mobility, context-awareness, and environment independence. The application framework layer of Mobile Gaia (Figure 1) provides a framework that automates the identified patterns and facilitates the creation and management of ubiquitous applications and their components. The application layer extends the Model-View-Controller paradigm [13], and facilitates the decomposition of an application into different input elements, output elements, and a model, where the logic of the application resides. This layer makes it easy for application developers to develop distributed applications that run on different devices in the personal active space. The framework provides a common interface for deploying and managing distributed applications. By utilizing the automated common programming patterns, developers can incorporate multi-device support, dynamic adaptation, mobility, and context awareness seamlessly into their applications.

VI. SAMPLE APPLICATION

In this section we discuss a consumer-level Mobile Gaia application that we are designing. This application is an Indoor/Outdoor Navigation System that uses the Mobile Gaia middleware for locating a person in indoor as well as outdoor environments, providing directions to a point of interest and notifying the user when he enters a certain region of interest. The application is designed using our application framework. It consists of a model component that contains the application logic; Query Interface, Location Receiver and Trigger Receiver components that act as input elements of the application, and Spatial Display and Audio Tracker components that act as output elements.

Location information is gathered by two possible devices, a GPS device for outdoor positioning, and a cellular phone with RFID for indoor positioning. The Location Receiver component receives location updates from the location service of Mobile Gaia. It sends this information to the model. The model notifies the Spatial Display component to display the position of the object in its spatial map. Spatial Display component provides a visual representation of the spatial model stored in the location service of Mobile Gaia. It displays the location of the object on the map similar to a GPS display device. The Query Interface can be used to query for places of interest and also to map out a route to a destination. Trigger Receiver component receives location triggers when a certain spatial condition is satisfied. For example, if the user has programmed to be notified when he is close to an ATM machine, the location system sends a trigger when the user is within a certain distance from the ATM machine. The Audio Tracker component provides audio notifications. These components can be run on different devices and our application framework and services facilitate interactions.

VII. RELATED WORK

Wireless Personal Area Network (WPAN) [14] is a well-researched area where devices communicate over short-range wireless links. There are some WPAN systems based on Bluetooth and WiFi [15]. WPAN provides necessary services for discovery and interaction among devices. Mobile Gaia uses WPAN as a communication substrate and supports several middleware services. It also enables interaction among these services. We also support an application framework to build applications over Mobile Gaia middleware. Wang et al. [5] present a model for forming device groups based on context information. Central to this research work is the concept of a social group that identifies the devices that need to be formed part of the group. This project does not define a middleware for forming device groups and does not have an application framework to develop applications for this group. The Impromptu project [16] supports a framework for building ad-hoc pervasive systems. The framework provides a context communication language that facilitates communication among participating entities in an ad-hoc manner. The project does not support a generic service based middleware or application framework like Mobile Gaia.

VIII. Conclusion

In this paper we introduced our middleware, Mobile Gaia, for ad-hoc pervasive computing. Mobile Gaia facilitates the integration of resources of a cluster of personal devices and provides a common programming interface to program the cluster. Services are designed as components that can be dynamically loaded and unloaded from the middleware using a deployment framework. The middleware also supports an application framework that can decompose an application into components. This decomposition enables an application to be distributed across multiple devices. In this paper, we have described the architecture of Mobile Gaia and a sample application.

IX. References

[1] M. Roman, C. K. Hess, R. Cerqueira, R. H. Campbell, and K. Narhstedt, "Gaia: A Middleware Infrastructure to Enable Active Spaces," *IEEE Pervasive Computing Magazine*, vol. 1, pp. 74-83, 2002.

[2] C. D. Kidd, R. Orr, G. D. Abowd, C. G. Atkenson, I. A. Essa, B. MacIntyre, E. Myanatt, T. E. Starner, and W. Newstetter, "The Aware Home: A Living Laboratory for Ubiquitous Computing Research," presented at CoBuild'99, 1999.

[3] B. Johanson, A. Fox, and T. Winograd, "The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms," *IEEE Pervasive Computing Magazine*, vol. 1, 2002.

[4] M. Mutka and D. Zhu, "Promoting Cooperation Among Strangers to Access Internet Services from an Ad Hoc Network," presented at IEEE International Conference on Pervasive Computing and Communications, 2004.

[5] B. Wang, J. Bodily, and S. K. S. Gupta, "Supporting Persistent Social Groups in Ubiquitous Computing Environments Using Context-Aware Ephemeral Group Service," presented at IEEE International Conference on Pervasive Computing and Communications (PerCom'04), Orlando, FL, 2004.

[6] F. Kon, T. Yamane, C. Hess, R. Campbell, and M. D. Mickunas, "Dynamic Resource Management and Automatic Configuration of Distributed Component Systems," presented at 6th USENIX Conference on Object-Oriented Technologies and Systems (COOTS'2001), San Antonio, Texas, 2001.

[7] M. Roman, A. Singhai, D. Carvalho, C. Hess, and R. H. Campbell, "Integrating PDAs into Distributed Systems: 2K and PalmORB," presented at International Symposium on Handheld and Ubiquitous Computing (HUC'99), Karlsruhe, Germany, 1999.

[8] F. Kon, R. H. Campbell, M. D. Mickunas, K. Nahrstedt, and F. J. Ballesteros, "2K: A Distributed Operating System for Dynamic Heterogeneous Environments," presented at 9th IEEE International Symposium on High Performance Distributed Computing, Pittsburgh, 2000.

[9] R. W. Woodings, D. D. Joos, T. Clifton, and C. D. Knutson, "Rapid Heterogeneous ad hoc Connection Establishment: Accelerating Bluetooth Inquiry Using IrDA," presented at Wireless Communications and Networking Conference (WCNC 2002), 2002.

[10] A. Ranganathan, J. Al-Muhtadi, S. Chetan, R. Campbell, and M. D. Mickunas, "MiddleWhere: A Middleware for Location Awareness in Ubiquitous Computing Applications," presented at 5th International Middleware Conference (Middleware 2004) (accepted), 2004.

[11] UbiSense, "Local position system and sentient computing." http://www.ubisense.net/.

[12] M. Roman and R. H. Campbell, "Providing Middleware Support for Active Space Applications," presented at ACM/IFIP/USENIX International Middleware Conference, Rio de Janeiro, Brazil, 2003.

[13] G. E. Krasner and S. T. Pope, "A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System," *Journal of Object Oriented Programming*, vol. 1, pp. 26-49, 1988.

[14] T. Siep, I. Gifford, R. Braley, and R. Heile, "Paving the way for Personal Area Network Standards: An Overview of the IEEE P802.15 Working Group for Wireless Personal Area Networks," *IEEE Personal Communications*, vol. 7, pp. 37-43, 2000.

[15] T. Salonidis, P. Bhagwat, L. Tassiulas, and R. LaMaire, "Distributed Topology Construction of Bluetooth Personal Area Networks," presented at Infocom, 2001.

[16] M. Beigl, T. Zimmer, A. Krohn, C. Decker, and P. Robinson, "Creating Ad-Hoc Pervasive Computing Environments," presented at Second International Conference on Pervasive Computing, Linz/Vienna, Austria, 2004.