

# A Middleware for Enabling Personal Ubiquitous Spaces

Shiva Chetan, Jalal Al-Muhtadi, Roy Campbell, M. Dennis Mickunas

Department of Computer Science, University of Illinois at Urbana-Champaign  
{ chetan, almuhtad, rhc, mickunas }@cs.uiuc.edu

**Abstract.** Ubiquitous computing enables an environment that assimilates digital and physical devices seamlessly and presents a unified programming interface to the user. Users can program the environment similar to programming a computer. With the widespread availability of personal devices and personal area networks, there is a growing need for personal devices to share resources and services among themselves to support complex applications. Middleware support is required for enabling interactions among services and for sharing resources. We introduce Mobile Gaia, a middleware framework to enable the construction of personal ubiquitous computing environments, or personal spaces, which are formed ad hoc using personal devices carried or worn by a person as well as devices that are physically nearby. We discuss the architecture and services of Mobile Gaia and some of the challenges that need to be addressed in this endeavor.

## 1 Introduction

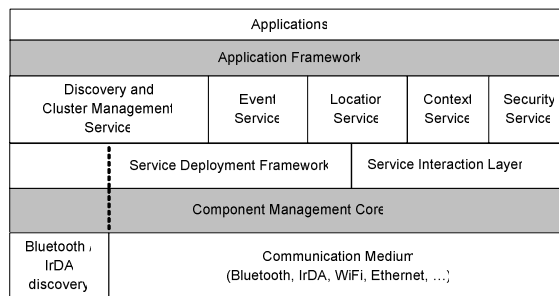
The traditional notion of ubiquitous computing focuses on building “smart” physical spaces populated with a plethora of embedded devices, sensors, actuators, and digital communication primitives. These physical spaces are then transformed into context-aware information-rich “ubiquitous computing environments,” which can take on the responsibility of serving users by tailoring themselves to users’ preferences and performing tasks and group activities according to the nature of the spaces and their contents and purposes. As a result, users can focus on the tasks at hand, minimizing distractions, and allowing machines to learn, adapt, and facilitate users’ interactions with their surroundings. With recent advances in mobility, miniaturization, wireless networking, and wearable computing, ubiquitous computing spaces are no longer confined to special spaces with physical boundaries. We envision a new domain where a person carries or wears various heterogeneous devices with different capabilities, including PDAs, digital cameras, smart phones, smart watches, wearable computers, and MP3 jukeboxes. The plethora of personal devices will be further enriched with wireless communication capabilities in the form of WiFi, Bluetooth, and/or IrDA.

In this paper, we introduce Mobile Gaia, a middleware framework to enable the construction of personal ubiquitous computing environments, which are formed ad hoc using personal devices carried by a person as well as devices that are in close proximity. We refer to a personal ubiquitous computing environment as a *personal space*. The middleware provides the necessary primitives to discover personal devices, bootstrap a personal space, and support large collections of heterogeneous personal devices. Since it is difficult to predict in advance the most appropriate application configuration in personal spaces, the middleware provides primitives for service and application deployment and management, where the services and applica-

tions become dynamic entities with multiple input, output, processing, and data storage alternatives that can change dynamically as needed. As people meet, interact, and collaborate, their personal spaces merge to provide better resource sharing and seamless interaction.

## 2 Mobile Gaia Architecture

Mobile Gaia consists of a set of services to maintain the cluster, facilitate event communication among devices in the cluster, provide location information to all devices and enable secure access to resources of the cluster. The architecture of Mobile Gaia is shown in Figure 1. Services in the middleware are implemented as components that can be dynamically loaded and unloaded. Component-based design has been found suitable for integrating resource-constrained devices into pervasive systems from our earlier research [1].



**Figure 1: Mobile Gaia Architecture**

Each device in the cluster contains the Component Management Core and Service Deployment Framework layers of Mobile Gaia. These layers enable the creation and deployment of components on the host device. Service interaction layer is used for inter-personal space communication. This layer facilitates integration of services across personal spaces.

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. The event service is a publish-subscribe system that allows applications to subscribe to events of interest. When the event occurs, the subscribers are notified by the event service. Location service provides location information to all devices in the cluster. A typical indoor location system can have several location-sensing technologies such as Radio Frequency badge detectors, Bluetooth and WiFi base stations and Ubisense [2]. Each device in the cluster can only be detected by a few location-sensing technologies. For example, a Bluetooth-enabled handheld with a WiFi capability can detect its location relative to a Bluetooth or a WiFi access point. Similarly, an RFID-enabled mobile phone with infrared technology can be detected with an RF or infrared signal detector. So the location service uses sensor-fusion techniques to combine location information from different sensing technologies and arrives at a probabilistic value for location of the devices [3]. 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.

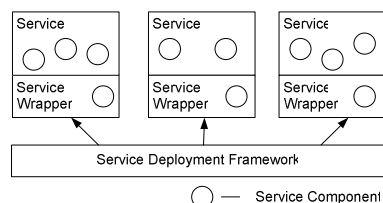
### 3 Service Management

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 the middleware.

#### 3.1 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 our previous model [4]. In our model, a system loads a minimal set of components required to provide a certain service to applications. This model was successfully used to design our middleware for infrastructure-based pervasive systems. We are reusing the model to integrate mobile devices into infrastructure-free distributed systems.

The Service Deployment Architecture is shown in Figure 2. 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, the location service in Mobile Gaia consists of a sensor fusion component, spatial model component and GPS-adaptor component. 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 laptop. Therefore, the location service of the laptop consists of the sensor fusion component, spatial model component and GPS-adaptor component. 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 present in the location service. The location service wrapper provides appropriate interfaces to the location



**Figure 2: 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. The service deployment framework also facilitates resource sharing.

One current drawback of our deployment architecture is that the coordinator device is expected to host all services in coordinator roles. This imposes heavy burden on the coordinator device in terms of resources such as memory, battery power and network

bandwidth. In addition, the coordinator device should provide platform support for services in coordinator role. We are investigating service deployment techniques to find optimal deployment patterns that efficiently use resources of all devices in the personal space. This is an on-going work and is discussed in Section 4

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.

## 3.2 Cluster Management Service

The cluster management service provides primitives to bootstrap a personal space, discover devices in the vicinity, and add/remove devices to/from the cluster as needed. At the bare minimum, any potential execution node in a personal space should be running the cluster management service, otherwise, the node will not be discoverable. Further, to prevent unauthorized resource sharing and thwart malicious devices from joining unauthorized spaces, it is necessary to perform some level of device/space authentication and authorization. The following is brief description of the bootstrapping and discovery primitives that this service provides.

### 3.2.1. Setting up a Personal Space

Before personal devices can form a personal space, there is a one-time setup configuration phase for the personal devices. For each personal space, a device or more can be designated as the coordinator device for space. 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 space. A key pair  $(PK_{\text{space}}, PK_{\text{space}}^{-1})$  is generated for this space. Key pairs are also generated for each device  $(PK_{\text{device}}, PK_{\text{device}}^{-1})$ .
2. On all client personal devices, a minimal version of the middleware is installed along with a list of configured with a list of spaces that they are permitted to connect to. For each space, the public key  $(PK_{\text{space}})$  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 full middleware, configure the space names that it is permitted to coordinate, and the public/private key pair  $(PK_{\text{space}}, PK_{\text{space}}^{-1})$  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.

### 3.2.2. Bootstrapping a Personal Space

Our aim is to provide an automated and secure method for bootstrapping a personal 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 solu-

tion that does not require users to reconfigure their devices each time a personal space is bootstrapped. The bootstrapping process works as follows.

1. The discovery service running on a coordinator machine tries to discover nearby devices.
2. Upon discovering a client device, an invitation is sent to join the space, along with the space name: Coord. → Client: <invitation, 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. Alternatively, the personal device may reject the invitation if the space name is unrecognized: Client → Coord.: <accept, random\_num>
4. The coordinator returns a packet containing its communication address and the random number signed with the space's digital signature: Coord. → Client: <address, signed{random\_num}<sub>space</sub>, random\_num2>.  
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: Client → Coord: <signed{random\_num2}<sub>device</sub>>
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.

### 3.2.3. 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.

## 3.3 Event Service

Event service facilitates event communication among devices in the cluster. The event service is a publish-subscribe system that decouples event senders from receivers. The coordinator role of the event service consists of the EventChannelWrapper, EventChannelRepository and EventChannelManager components. The EventChannelWrapper acts as an interface to the service components. EventChannelRepository stores information about event channels that are created in the cluster. Applications can query this repository to obtain handles to event channels. EventChannelManager provides interfaces to create, destroy, publish to and subscribe from event channels. In the client role, only EventChannelWrapper and EventChannelManager components

are loaded since EventChannelRepository runs on the coordinator device of the cluster.

### **3.4 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. Different devices in the cluster have different location-sensing technologies such as GPS, WiFi, Radio Frequency Identification and so on. Our location service fuses location information from different devices and arrives at a probabilistic estimate of the location information. 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.

### **3.5 Security Service**

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 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 keys, as described earlier. 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.

### **3.6 Inter-Personal Space Interactions**

Mobile Gaia also supports service interactions across personal spaces. Such interactions are useful when resources of different spaces need to be shared. Inter-Personal Space Interactions enable a host of applications such as “ad-hoc” meetings among personal space users, secure file-sharing across personal spaces, applications requiring ad-hoc service compositions and so on.

Mobile Gaia has a service interaction layer to facilitate inter-personal space interactions. This layer enables integration of services across spaces. For example, the Cluster Management Service maintains a repository of all devices present in the personal space. Applications and other services can query this repository to obtain information about these devices. When two personal spaces are integrated, queries to the Cluster Management Service are routed to the Service Interaction Layer by the ser-

vice. The interaction layer forwards these queries to corresponding services of the other space. The results of these queries are collected by the service interaction layer and sent back to the Cluster Management Service. The Cluster Management Service sends the combined result to the querying application.

The service interaction layer provides separation of concerns by managing interactions with services in other spaces. Without the service interaction layer, services had to provide additional functionality for inter-personal space interactions. We identified certain common patterns of interactions among services of the middleware and this motivated the creation of a separate layer that implements these patterns. Inter-personal space interactions in the middleware fall under three broad categories. (1) Distributed Querying, (2) Distributed Events, and (3) Direct Access.

In the distributed querying pattern, certain queries to local services are forwarded to services in other spaces. These queries include space-independent resource, component and sensor information queries. Distributed event pattern is used for event communication across spaces. These events include location and sensor-based triggers and triggers generated when devices are discovered. Location, discovery and context services use the above two patterns. In direct access pattern, a service sends a query to the service interaction layer specifying the space that it needs to query information from. The interaction layer forwards this query to the service in the specified personal space. Security and event services use the direct access pattern. Services use multiple patterns depending upon the functionality that they provide. The service interaction layer implements these patterns and additional patterns can be added to this layer if required.

## 4 Application Framework

Mobile Gaia provides an application framework that decomposes an application into multiple components. These components can be run on the same or different devices in the personal space. The decomposition also facilitates fine-grain adaptation of the application to the resources present in the personal space. We discuss a sample application that runs on a personal space using Mobile Gaia.

Certain applications can benefit by recording the location of mobile entities at given points in time. These include location-based chat, location-annotated photography and so on. In these applications, a “location-snapshot” of all concerned mobile objects is recorded at a particular instant. This enables application users to later relate the application contents with the recorded location information. We call this process of taking a location snapshot as *location-stamping*.

Mobile Gaia enables location-stamping in environments with heterogeneous location technologies. A typical location-stamping enabled photography application records the location of its concerned subjects when a photograph is taken. The location information can be later used to relate the location of the subject to the context of the photograph. Our application framework decomposes the application into one input component – ImageCapture, one output component – DisplayImage, one storage component – StoreImage and one logic component, LocationPhotograph, that coordinates the functioning of the other components. The ImageCapture component runs on a smart camera device, DisplayImage and StoreImage components run on a handheld device and LocationPhotograph component runs on the user’s laptop. When a photograph is taken by the camera, the ImageCapture component sends a digital copy of the

photograph to the LocationPhotograph component. On receiving the image, the LocationPhotograph component queries the location service about the location of concerned subjects. This information along with the digital copy of the photograph is sent to the StoreImage component that stores this information.

## 5 Conclusion and Future Work

In this paper we introduce a middleware framework to enable the construction of personal spaces, which are formed ad hoc using personal devices carried or worn by a person. The middleware provides the necessary services to bootstrap and manage personal spaces, share resources, and deploy services. The middleware also facilitates inter-space collaboration and resource sharing.

Mobility of nodes presents several challenges at the application/service layers that should be addressed by the ubiquitous computing community. One challenge we are exploring is automated and optimized service deployment in personal spaces. In effect, a personal space is a distributed system containing various devices. These devices have diverse capabilities such as memory, battery power, communication bandwidth, and so on. When deploying a service in a personal space, components of the service can be deployed on different devices based on the component requirements and device capabilities. The service can provide hints to the deployment framework to guide the deployment process. For example, a spatial model component requires large amount of memory but medium communication bandwidth to respond to spatial queries. Therefore, the spatial model can be deployed on a laptop with adequate memory and a Bluetooth connection. Similarly, other quality of service requirements can be used to guide the deployment process. This process can be modeled as a constraint-satisfaction problem where service requirements are treated as constraints. In addition to QoS requirements, user preferences need to be considered when deploying a service. A user may prefer to interact with his personal space using only his cell phone. User preferences should be factored into the constraint-satisfaction problem.

## 6 Related Work

Wireless Personal Area Networking (WPAN) [5] is attracting attention in the ubiquitous computing community with the introduction of short-range wireless communication technologies. WPAN provides necessary services for discovery and interaction among devices. Mobile Gaia uses WPAN as a communication substrate and supports several middleware services. The focus of Mobile Gaia is to orchestrate the functioning of services across various devices. Wang et al. [6] 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 [7] 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.

Several middleware architectures for infrastructure-based pervasive systems such as Gaia [8], IROS [9], Roomware [10], and Aware Home [11] have been developed. Infrastructure-based pervasive systems do not face issues such as ad-hoc network management, resource and service management in ad-hoc environments and related security issues. We have reused some of the concepts that we introduced in the Gaia project to design Mobile Gaia.

## 7 References

- [1] 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.
- [2] UbiSense, "Local position system and sentient computing." <http://www.ubisense.net/>.
- [3] 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.
- [4] 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.
- [5] 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.
- [6] 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.
- [7] 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.
- [8] 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.
- [9] B. Johanson, A. Fox, and T. Winograd, "The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms," *IEEE Pervasive Computing Magazine*, vol. 1, 2002.
- [10] N. Streitz, J. Geissler, and T. Holmer, "Roomware for Cooperative Buildings: Integrated Design of Architectural Spaces and Information Spaces," presented at Workshop on Cooperative Buildings (CoBuild'98), Darmstad, Germany, 1998.
- [11] 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.