Journal of Web Engineering, Vol. 6, No.2 (2007) 131-142
© Rinton Press

A WEB SERVICES BASED ARCHITECTURE FOR SUPPORTING MOBILE USERS IN LARGE ENTERPRISES

ANTONIO CORONATO and GIUSEPPE DE PIETRO

Istituto per il Calcolo e le Reti ad Alte Prestazioni

Consiglio Nazionale delle Ricerche

Email: {antonio.coronato, giuseppe.depietro}@na.icar.cnr.it

Received March 20, 2006
Revised September 20, 2006

Business enterprises can proficiently benefit from Ubiquitous Computing (UbiComp). UbiComp environments enable mobile users to access services and resources in a transparent and spontaneous way, that is, without any configuration operation and through very friendly user-environment interface mechanisms which sometimes anticipate user needs. Unfortunately, until now the UbiComp model has been applied only to environments bounded to a specific site. In order to support large enterprises, which may have several offices spread over a wide geographic area, this model needs to be extended with basic services that enable the realization of federations of environments. In other words, in the case of multi-office enterprises, each office could be equipped with a classic UbiComp environment; however, such environments must be interconnected by the internet and co-ordinated by a set of higher level services. As a matter of fact, in this case mobile users not only have to get access in a transparent and spontaneous way but, also, they should have the possibility of leaving a physical site without concern of their pending computations and, successively, they should be enabled to come back in the environment, even in a different site, and resume their computations. As a consequence, locating mobile users and handling their disconnections becomes a very critical requirement. Indeed, differently from a classic UbiComp environment, in a federation of UbiComps resources can not be merely released when a mobile user leaves. Instead, the global environment has to infer user's intentions in order to understand whether he will come back to resume computations, or not. In this paper, we propose a Web Services based architecture for federating classic UbiComp environments. This architecture supports large enterprises with functionalities for locating and tracking mobile users in a federation of UbiComp environments. These services implement specific strategies that have been devised to allow i) mobile users to leave a physical site with no concern of their pending computations; ii) the global environment to locate and track users in the federation of UbiComp environments; iii) mobile users to come back in the federation and resume their computations and iv) the environment to reliably handle resources by inferring users intentions.

Key words: Web Based Federations of Ubiquitous Computing Environments, Location and Tracking Services, Reliable Resources Handling

Communicated by: M. Gaedke & P. Fraternali

1 Introduction

Technology evolution in hardware design and development (including microprocessors, network bandwidth, and wireless devices), which has led to a dramatic improvement in terms of performance to cost ratio and of the integration scale factor, is leading towards the implementation of the Mark Weiser vision [1]. The emerging computing model is called *Ubiquitous Computing*. The ultimate goal is the development of environments where highly heterogeneous hardware and software components can seamlessly and spontaneously interoperate, in order to provide a variety of services to users independently of the specific characteristics of the environment and of the client devices [3]. Therefore, mobile devices should come into the environment in a natural way, as their owner moves, and transparently, that is, the owner should not have to carry out manual configuration operations for being able to approach the services and the resources.

Current realizations of Ubiquitous Computing environments are characterized by being bounded to a unique physical site [11,12,13]. Such environments are participated by a set of users, hardware and software components, which is highly dynamic and cannot be predicted in advance. As a consequence, the sudden departure or arrival of a service, device, or user should be considered normal operation, not an exceptional condition or a failure requiring special handling [5].

In the next generation of UbiComp environments, the computation model has to scale to large enterprises, which might have different offices distributed over a wide area, in different physical sites. In that case, the environment has to offer a unique interaction model to its users. The user has to perceive a unique, uniform, virtual environment, even though it consists of several physical environments. However, for this kind of environments, the list of available services may depend on the physical site the user is in (as an example, a video conference can be produced only in sites equipped with video cameras and streaming systems). Moreover, users could leave the virtual environment while having pending computations. In this case, the environment has to decide whether to let computations go on, or to suspend them and free their allocated resources.

These characteristics call for advanced location and tracking mechanisms. In particular, basic services are required for determining mobile users in the environment and their location. In addition, the environment must be able to track user movements and activities. Actually, since active devices affect the environment in terms of in use/available resources and services, we can conclude that the environment must reliably handle resources and services when users move around. It must be able to predict whether or not the user will require to resume its computation later. Then, active resources must be kept allocated, or freed, depending on what the user is supposed to do.

This paper describes an infrastructure of basic services for reliably handling mobile users sessions in a wide-area Ubiquitous Computing Environment.

The rest of the paper is organized as follows. Section 2 presents some motivations and related work. Section 3 describes the strategies and the services functionalities. Section 4 presents some implementation details. Section 5 describes a real scenario. Section 6 concludes the paper.

2 Motivations and contribution

The final objective of this work is the application of the Ubiquitous Computing model to large enterprises that have several affiliated offices and sites, physically dislocated over far distance each other. Each site could be equipped with a classic UbiComp environment that would grant access to mobile users in a transparent and spontaneous way, providing them with all the facilities that the UbiComp model can assure. However, by simply doing that, such UbiComp environments would result to be disconnected from each other and no correlation on resources and services deployed in different sites would exist even though they are part of the same organization.

It is important to note that a similar problem has attracted the interest of another scientific community, the Grid Computing one [3]. Grids are geographically distributed environments, equipped with shared heterogeneous services and resources accessible by users and applications to solve complex computational problems and to access big storage spaces. Grid applications are topologically organized to compose intra-Grids, extra-Grids or inter-Grids. Intra-Grids pool resources offered by different divisions existing within a single organization, which defines policies to share, access and use such resources. Computers of a single organization use a common security domain and share data internally on a private/LAN network. Extra-Grids bring together more intra-Grids using a remote/WAN connectivity. They are based on a single large organization or on multiple organizations, characterized by more than one security domain, which cooperate among them through external partnerships in order to address business, academic and other topics. For this reason, these partners have to agree common policies, in addition to or extending the existing single organization ones, to enable the access to services and resources of the extra-Grid. Finally, inter-Grids are based on many organizations distributed in the world, which pool their own resources using an Internet/WAN connectivity in order to sell, for example, computational or storage capacity.

With respect to such a classification, current implementations of UbiComp environments can be compared to Intra-Grids. As a matter of fact, UbiComp environments are LAN based applications that enable mobile users to access resources and services, which are deployed in a specific physical site. Users must be in the site in order to interact with the environment; that is, interactions from other sites are not allowed.

The target for this work is the realization of federations of UbiComp environments that behave as an extra-grid; that is, the federation has to enable users to access resources and services from different physical sites and, in addition, must allow them to be nomadic among sites. Thus, it is important to clarify why we do not apply the Grid model. We aim at supporting large business enterprises, which typically want their employees be supported by mobile and wireless technologies and by user friendly environments and services to access databases and other enterprise's applications in the easiest possible way. On the other hand, instead of business applications, the Grid model has been devised to provide support for high performance computing. As a consequence, the Ubiquitous Computing model has appeared to be more suitable for such kind of applications. But, we have already pointed out the limitations of such a model when applied to large enterprises. In other words, we need to build federations of UbiComp environments in order to allow that i) mobile users perceive a unique environment; ii) they access services independently of the physical site they are in; and, iii) they also move among sites and be granted about the computations they have launched. In this scenario, the major difficulty for the environment is just interpreting user intentions when a disconnection occurs. As a matter of fact, when a disconnection is detected, it is possible that:

- The user has definitively left the environment – He is no longer interested in pending computations (he just leaves the environment in a transparent way, that is without concerning about pending computations);
- User's device has temporarily or permanently failed;
- The user has turned off his mobile device – He is interested in computation results but he wants a break or to save battery while waiting for them;
- The user is moving in a "black" zone, that is an area of the physical site not covered by wireless connectivity;

- The user has left a site – He may or may not be interested in pending computations; he may return in the environment, even in a different site, and be willing or not to resume his working sessions.

Obviously, all these events manifest themselves to the environment identically; that is, as a mobile user disconnection. However, as already pointed out, such events require different behaviour for the environment.

In a previous work, we focused on the problem of locating and tracking mobile users in a federation of UbiComp environments [18]. In particular, a prototype service was presented.

In this work, the contribution consists in an architecture of services that enables to realize a federation of UbiComps and reliably supports mobile users in the virtual environment. In particular, strategies have been devised and implemented by services for i) granting access to mobile users, in a transparent way, independently on the physical site they are in; ii) locating and tracking them while in the federation of environments; iii) handling their disconnections in a reliable way; iv) handling their mobile sessions; and iv) reasoning on future user intentions whenever he disconnects from the environment. In addition, an architectural model for federating UbiComp environments has been designed. The architecture consists in a set of basic services that implements the proposed strategies.

3 Strategies and mechanisms for handling mobile users

In this section, we present the strategies that we adopted for realizing a reliable federation of UbiComps.

3.1. Connecting incoming mobile users

The mechanism we chose for providing network connectivity on-the-fly is based on the Dynamic Host Configuration Protocol (DHCP) [8], which is also a well known solution for the implementation of basic location functions [7]. DHCP dynamically assigns an IP address to an incoming device, which is, then, able to access to the network. It can be used both for 802.11b and for Bluetooth enabled devices in the PAN profile or in the LAN Access Profile [14]. No particular assumptions about the client device must be taken except that it must be configured as a DHCP client.

It's worth noting that standard DHCP protocol has not been devised for highly dynamic environments. As a matter of fact, the IP address assigned to a device is locked until the LEASE_TIME expires. The LEASE_TIME is a parameter that typically varies from 12 to 24 hours. During this time, standard DHCP protocol doesn't take care about possible early user disconnections. Some limitations of the standard DHCP protocol for mobile computing environments were already pointed out in [9] and are not faced by the forthcoming DHCP RFC [10], which introduces the possibility of forcing the mobile device to renew the IP request once the lease time expires. However, UbiComp environments have different requirements. Indeed, when a mobile device disappears from the environment, it continues to have an associated IP address until its LEASE_TIME expires, making such a network resource unavailable for incoming new devices. For this reason, some additional functionalities have been developed. In particular, we made the DHCP server able to free IP addresses on demand. By doing so, the environment can require to release network resources (IP addresses) when mobile users leave. It's also important to note that such additional functionalities do not affect the DHCP protocol and DHCP clients are just standard clients.

The enhanced DHCP service that we developed exposes the following additional functionalities:

- *releaseIP* – This operation releases the IP address previously assigned to a leaving device;
- *NEW_DEVICE* – This event is forwarded to the environment whenever an incoming device requires an IP address.

3.2. Locating mobile users

In the environment, at any time, a variable number of devices may be active. These devices can be located in different sites and even in different locations within the same site. The environment has to locate them in order to provide customized services. In general, in an UbiComp environment both location and locating functions must be available. A location function is a mechanism for identifying objects active at a specific location, whereas a locating function is a mechanism for identifying the location of specific objects. As an example, our environment uses the location function for determining devices active within a multimedia room and providing them with access to an *ETestingService*. Differently, when a mobile user requires to print a document, the environment uses the locating function to locate him and the nearest printer.

In our federation of UbiComp environments, each site is composed of different locations that correspond to physical areas covered by distinct wireless Access Points (APs), i.e. each AP identifies a physical area in the site. As a consequence, knowing objects location is a two level problem. The first level consists in determining the physical site, whereas the second level consists in determining the physical area within the site. Locating and location functions are exposed by a *LocationService*, which supports the following operations and asynchronous communications:

- *localizeObject* – This operation returns the location of a specific device;
- *getObjects* – This operation returns a list of active devices to a specific location;
- *NEW_LOCATION* – This event is dispatched to the environment whenever an active device changes its position within the environment, i.e. it has moved into a new location.

In order to locate devices in the environment, the *LocationService* periodically interrogates the wireless APs. Each AP writes an event into a log file whenever a device becomes active into its area. By comparing such logs and by handling global states, the *LocationService* is able to detect location changes. A similar approach has been realized in [15]. It is also worth noting that a more effective location mechanism can be obtained by equipping environments with active location systems like those described in [6]. However, active location systems require that a software element be installed on board the mobile device. Therefore, this condition limits the capability of a generic mobile device to interact with the environment in a transparent way since it requires a preliminary operation of software installation.

3.3. Detecting user disconnections

To recognize user disconnections an *EcoService* has been developed. The *EcoService* periodically detects each mobile device with a ping operation. After having issued a ping message the *EcoService* waits for a response or for a timeout. A mobile device is declared inactive after having missed a certain number of consecutive ping messages. Ping intervals, as well as the number of missing ping responses needed to declare inactive a mobile device, are parameterized and can be changed dynamically. When a mobile device is no longer active in the environment, the *EcoService* notifies the event DEVICE_INACTIVE.

3.4. Tracking user movements

In the environment, user movements are caught by a *TrackingService*. The *TrackingService* has been built on top of the *LocationService*, the *EcoService* and the *DHCPService*. Indeed, it collects events like NEW_DEVICE, NEW_LOCATION and DEVICE_INACTIVE.

3.5. Tracking user activities

The *TrackingService* has also been augmented with functionalities for tracking users activities. For such an aim, the following operations are supported:

- *ServiceStarted* – This operation is invoked when a mobile user requires a service;
- *ServiceCompleted* – This operation is invoked when a service completes or is closed.

Generally, a user can require more than one service in a session (issues related to Web Services Orchestration and Choreography can be found in [19]).

In the environment services are classified as Lowly Interactive Service (LIS) or Highly Interactive Service (HIS). LISs are services that need interactions with the user only in the preliminary stages to which long computational processes follow. On the other hand, HISs are services that require interactions with the user all along their processing. This classification enables to have some inferring policies. Indeed, the basic idea is that when a mobile user leaves the environment while having an HIS operating, probably he is no more interested in that service then it can be released. Differently, if the user leaves after having launched a LIS, he is supposed to be back to pick up results.

The state machine depicted in figure 1 enables the *TrackingService* to keep track of the activities of a mobile user. State transitions are forced by events as i) the user requires a service; ii) the user explicitly terminates a service; iii) the user leaves the environment; or, iv) a timeout expires.

In detail, we have two kinds of timeouts:

- *LIS_timeout* – As described previously, a mobile user can leave the environment after having launched a LIS service. In this case, the idea is to allow the service to continue its processing until results are obtained. After that, the environment waits for the user back or for a timeout before releasing resources and removing results. The *LIS_timeout* is the time that the environment waits before removing results and releasing resources.
- *HIS_timeout* – In the case the user becomes unavailable in the environment while having an HIS service active, *HIS_timeout* is the time that the environment waits before releasing the HIS.

The state machine has the following states and macro-states.

- *BackgroundComputing* – In this macro state only LISs are active.
- *ForegroundComputing* – In this macro state at least one HIS is active.
- *NoComputing* – In this macro state neither LISs nor HISs are active, but the mobile device is active in the environment.
- *LIS* – In this state at least one LIS is operating. No HIS is active.

- *BC_WAIT* – In this state the active LIS has completed its computation and waits for either the client picks up results or the timeout expires.
- *HIS* – In this state at least one HIS is active. No LIS is active.
- *LIS_HIS* – In this state at least one HIS and one LIS are active.
- *FC_WAIT* – In this state at least a HIS is active and all active LISs have completed their calculus and wait for either user picks up results or the timeout expires.

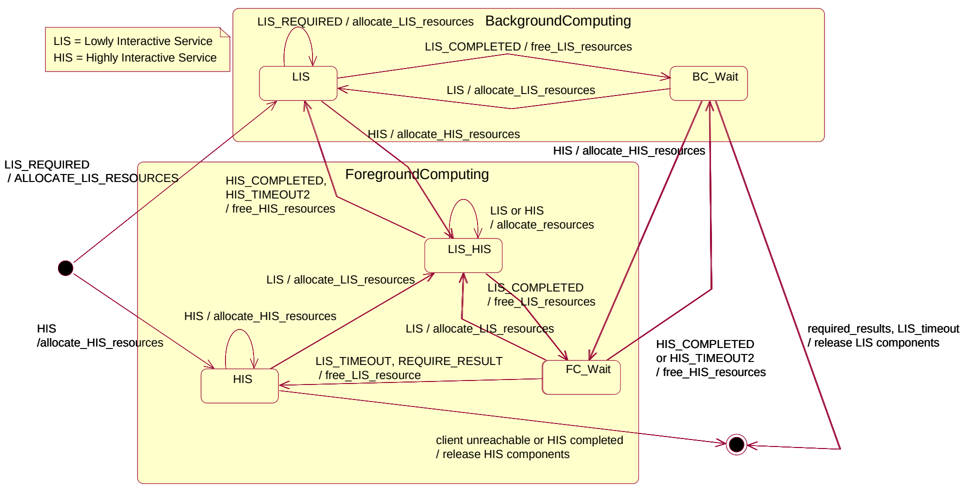
LIS = Lowly Interactive Service
HIS = Highly Interactive Service
LIS_REQUIRED / allocate_LIS_resources
BackgroundComputing
LIS_COMPLETED / free_LIS_resources
LIS
BC_Wait
LIS / allocate_LIS_resources
HIS / allocate_HIS_resources
HIS / allocate_HIS_resources
LIS_REQUIRED
/ ALLOCATE_LIS_RESOURCES
HIS_COMPLETED,
HIS_TIMEOUT2
/ free_HIS_resources
ForegroundComputing
LIS or HIS
/ allocate_resources
LIS_HIS
LIS / allocate_LIS_resources
LIS_COMPLETED
/ free_LIS_resources
HIS / allocate_HIS_resources
HIS
/allocate_HIS_resources
LIS / allocate_LIS_resources
required_results, LIS_timeout
/ release LIS components
HIS_COMPLETED
or HIS_TIMEOUT2
/ free_HIS_resources
LIS_TIMEOUT, REQUIRE_RESULT
/ free_LIS_resource
HIS
FC_Wait
client unreachable or HIS completed
/ release HIS components

Figure 1 - Client state machine

3.6. Inferring user intentions

The *TrackingService* is also able to infer user intentions. In particular, user intentions can be inferred from the state machine reported in figure 1. Accordingly to what established with the previous strategy, for a leaving user, who is in the LIS, BC_WAIT, LIS_HIS, or FC_WAIT state, it is assumed that he will come back later to pick up results. In all other states, user is supposed to not come back.

3.7. Disconnecting leaving users

In order to reliably disconnect leaving users, the *TrackingService*, which bases on user's inferred intentions, issues the following set of signals:

- FREE_ALL_RESOURCES – This event requires that all the allocated resources be freed;
- FREE_HIS_RESOURCES – This event requires that any HIS service is released;
- FREE_LIS_RESOURCES – This event requires that any LIS service is released. In particular, this event is issued when the last timeout for the waiting LIS services expire;
- FREE_NET_RESOURCES – This event requires that the network resources be released.

4 Implementation details

The virtual environment has been realized by means of a software infrastructure that integrates and co-ordinates different UbiComps. Each site has a set of basic services. Among such physical environments, one is chosen as the leader site. The leader site has additional responsibilities (and services) for co-ordinating all other UbiComp environments.

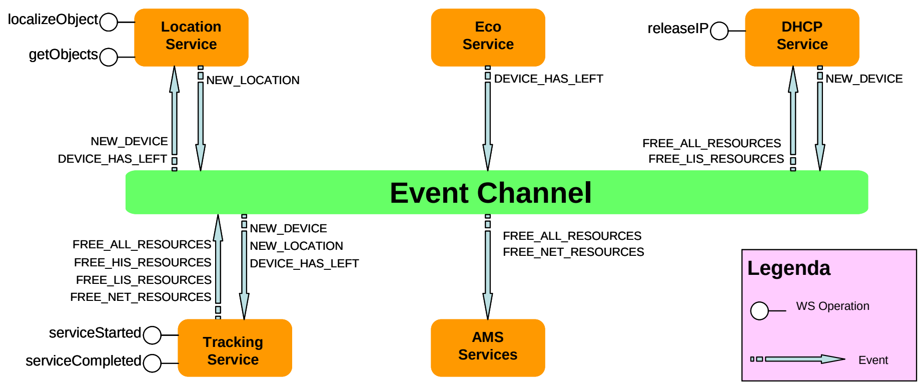

Figure 2 - Asynchronous communications and WSDL interfaces

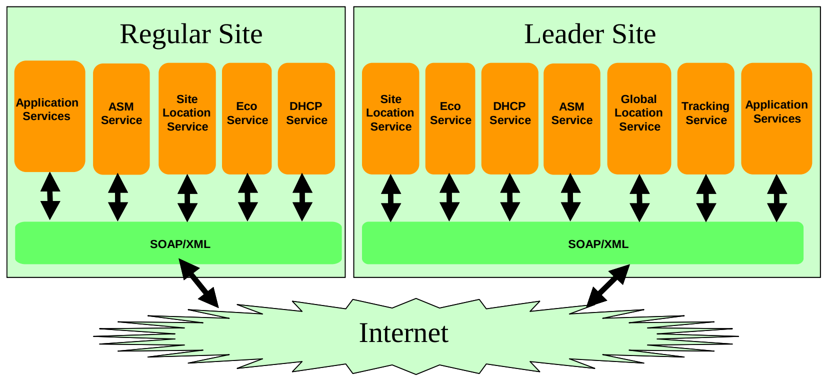

Figure 3 - Services deployment

The architecture consists of services that have been built as Java distributed objects and partially exposed as Web Services. In particular, the following services have been realized:

- *DHCPService* – This service is in charge of supporting the connecting strategies for incoming devices. In detail, it consists in an enhanced DHCP server that provides network access to mobile devices and releases IP addresses on demand. Each UbiComp environment is equipped with a *DHCPService*.
- *EcoService* – This service is in charge of detecting leaving mobile devices. It periodically issues ping messages towards mobile devices in order to establish whether the device is still available in the environment or not. In the case the device misses a certain number of consecutive messages, it is declared disconnected. Each site is equipped with an *EcoService*.

- *LocationService* – This service is in charge of locating mobile devices within the federation of environments and within each physical site. Therefore, the service consists in two levels, namely the *SiteLocationService* and the *GlobalLocationService*. A *SiteLocationService* is installed in each physical site and is in charge of locating mobile devices within it. Location information are notified to the *GlobalLocationService*, which is active only in the leader site and is then in charge of handling global location state.
- *TrackingService* – This service keeps track of user movements within the virtual environment. Therefore it collects location information coming from the location service. In addition, it handles a state machine (as depicted in figure 1) for each user. As a consequence, this service receives information about services and resources in use by the user and successively infers his intentions when he disconnects from the environment. The service is deployed only in the leader site.
- *ApplicationServiceManager* (*ASM*) – This service handles user requests for application services and successively communicates them to the *TrackingService*. The service is deployed only in the leader site.

As shown in figure 2, services use two communication mechanisms, the SOAP synchronous rpc exposed by their WSDL interfaces, and an asynchronous mechanism offered by an event channel. We implemented the event channel [16] by extending the WSBrokeredNotification [17] that specifies a subscribe-notify mechanism for Web Services. Finally, services are deployed as reported in figure 3.

5 Experimental scenario

The experimental environment consists of two physical sites. Site 1 is located in Naples. It is equipped with a printer and other resources. Site 2, which is the leader site, is located in a suburb quarter, a few kilometres far from site 1. It is equipped with a multimedia laboratory and some multimedia rooms. In detail, in site 2 the following resources are available:

- Motion Capture System – This is a system for capturing human motions. It consists of several cameras, which capture movements performed by an actor, and a graphic station, which reproduces movements as a skeleton accordingly with actor’s motions;
- Rendering Station – This is a workstation for rendering row motion data in 3D graphic applications;
- Projector – This is a projector, driven by a pc, to project multimedia presentations;
- Streaming Server – This server hosts a video streaming application;
- E-Testing Server – This server hosts an E-Testing application;
- Printers and other resources.

The environment supports the following application services:

- *MotionCaptureService* – This service drives the motion capture system. An actor (equipped with optical markers) moves around in the multimedia laboratory. Several cameras capture his movements that are reproduced on the graphic station. The graphic station shows a skeleton, which moves accordingly with the actor, and records data movement in a file. This service must be available only in site 2 because it requires that the actor move in the physical capture area of the system;

- *RenderingService* – This service enables users to submit row motion data and to build 3D graphic applications. This is a computational service that executes processes over the Rendering Station. However, it should be available in both sites. Indeed user can submit the input row motion file and choose the rendering options via a dialog form; successively, he can take up the output rendered file at the end of the job. Thus, although computation is performed by the rendering station, which is available in site 2, input file can be submitted from any remote place;
- *PresentationService* – This service enables a user to show its presentation in the multimedia room. The service receives a pdf/ppt file via a dialog form and then enables speaker to control the presentation flow. This is an interactive service, which requires the speaker to be in the room for presentation. As a consequence, the service must be available only in Site 2;
- *VideoConferenceService* – This service enables attendees to follow a presentation on their personal mobile device. A video server captures a presentation with its videocam and streams it over the network. The service isn't interactive. Users receive presentation images, but do not interact with the speaker. This service must be available in both sites;
- *ETestingService* – This service enables to perform on-line evaluation tests for courseware activities. When a session test starts, students must be in the multimedia room. Evaluation tests are synchronized and students have a predefined period of time for completing each test section. Students can interrupt their test by explicitly closing the service or by leaving the multimedia room. This service must be available only in the multimedia room of site 2;
- *PrintService* – This service enables users to print their pdf documents. It must be available in both sites. However, the print service must use the local printer of the site in which user requires printing.

Figure 4 shows the environment, the allocated resources, and the available services.

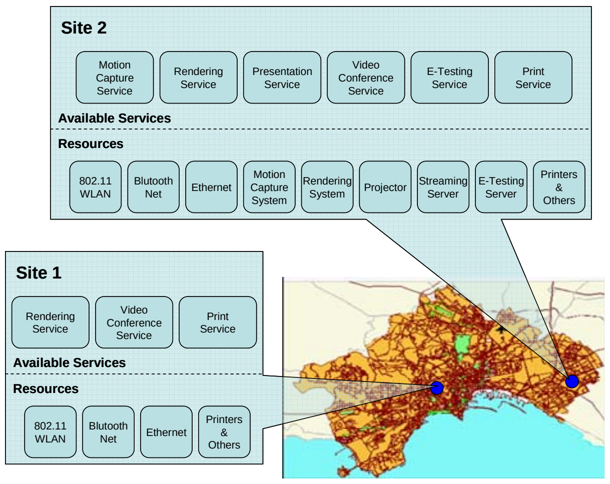

Figure 4 Environment overview.

In our scenario, application services are classified as HIS or LIS as reported in table 1.

Service	**Required Resources**	**Kind of Service**	**Availability**
MotionCaptureService	Motion Capture System	HIS	Site2
RenderingService	Rendering Station	LIS	Site 1, Site 2
PresentationService	Projector	HIS	Site 2
VideoConferenceService	Streaming Server	LIS	Site 1, Site 2
E-TestingService	E-Testing System	HIS	Site 2 (multimedia room only)
PrintService	Printer	LIS	Site 1, Site 2

Table 1 – Services

In this environment, when a new device enters, it dynamically obtains an IP address from the *DHCPService*. The *DHCPService* issues a NEW_DEVICE event that is caught by the *LocationService* and by the *TrackingService*. From this moment on, any service request/closure performed by the mobile user causes that the *ApplicationServiceManager* requires that the *TrackingService* update their data structures and state machines.

In the case the mobile user changes its position within the environment, the *LocationService* issues a NEW_LOCATION event.

When the mobile device becomes inactive, the *EcoService* issues a DEVICE_HAS_LEFT event. This event is handled by the *TrackingService* that infers on user intentions. In particular, by using the personal state machine, the *TrackingService* determines which resources and services must be released and consequently notifies an event accordingly with the disconnecting strategy described in section 3.6.

6 Conclusions

The ubiquitous computing model proved to be a promising computing model for realizing pervasive environments that make users interactions easier and more spontaneous. However, such environments are still far from being able to effectively and reliably handle the high degree of dynamicity that characterizes them.

In this paper we presented a novel model of UbiComp environments that can be applied to wide area organizations. In such a scenario, handling user mobility becomes even a more critical task for which we purposely devised strategies and developed some services that suitably add dependability characteristics to the environment.

References

1. M. Weiser, “The Computer for the 21st Century”, Scientific AM, September 1991, reprinted in IEEE Pervasive Computing, Jannuary-March 2002.

2. K. Lyytinen and Y. Yoo, “Issues and Challenges in Ubiquitous Computing”, Communications of ACM, December 2002, Vol. 45, N.12.

3. I. Foster, C. Kesselman, “The Grid: Blueprint for a New Computing Infrastructure”. Morgan Kaufmann, 1999.

4. D. Saha and A. Murkrjee, "Pervasive Computing: A Paradigm for the 21st Century", IEEE Computer, March 2003.

5. T. Kindberg and A. Fox, "System Software for Ubiquitous Computing", IEEE Pervasive Computing, January-March 2002.

6. J. Hightower and G. Borriello, "Location Systems for Ubiquitous Computing", IEEE Computer, August 2001.

7. G. Banavar, J. Beck, E. Gluzberg, J. Munson, J. Sussman, and D. Zukowski, "Challenges: An Application Model for Pervasive Computing", in the proc. of the 6th ACM/IEEE Int. Conference on Mobile Computing and Networking, MOBICOM 2000.

8. R. Droms, "Dynamic Host Configuration Protocol", RFC 2131, Internet Engineering Task Force, www.ietf.org.

9. C. E. Perkins and K. Luo, "Using DHCP with computers that move", Wireless Networks, 1995, pp 341-353, Baltzer AG, Science Publisher.

10. Y. T'Joens, et all, "DHCP Reconfigure Extention", RFC 3203, Network Working Group, www.ietf.org.

11. http://cobra.umbc.edu/

12. http://oxygen.lcs.mit.edu/

13. Garlan, D., Siewiorek, D., Smailagic, A., Steenkiste P., "Project Aura: Toward Distraction-Free Pervasive Computing", IEEE Pervasive Computing, April-June 2002

14. M. Albrecht, M. Frank, P. Martini, M. Schetelig, A. Vilavaar, A. Wenzel, "IP Services over Bluetooth: Leading the Way to a New Mobility", in the proc. of the 24th conference on Local Computer Networks, LCN 1999.

15. S. G. M. Koo, C. Rosenberg, H. H. Chan, Y. C. Lee, "Location-Based E-Campus Web Services: From Design to Deployment", in the proc. of the first IEEE International Conference on Pervasive Computing and Communications, PerCom'03.

16. M. Ciampi, A. Coronato, G. De Pietro and M. Esposito, "Handling Structured Classes of Events in Pervasive Grids", in the 4th International Metainformatics Symposium, MIS 2005, to appear as Lecture Notes in Computer Science, LNCS, Springer Verlag.

17. http://www.oasis-open.org/committees/download.php/13485/wsn-ws-brokered_notification-1.3-spec-pr-01.pdf

18. A. Coronato and G. De Pietro, "Locating and Tracking for a Meta-ubiComp Environment", in the 4th International Metainformatics Symposium, MIS 2004, Lecture Notes in Computer Science, LNCS, Springer Verlag.

19. http://devresource.hp.com/drc/technical_articles/wsOrchestration.pdf