International Journal of Fuzzy Logic and Intelligent Systems, vol. 5, no. 3, September 2005 pp. 189-195

Concord: A Proactive Lightweight Middleware to Enable Seamless Connectivity in a Pervasive Environment

Sam Hsu*, Mahesh Mutha, A.S. Pandya* and YoungUhg Lho*****

*** Dept. of Computer Science and Engineering, Florida Atlantic University, USA**
**** Motorola Inc., Plantation, Florida, USA**
***** Dept of Computer Education, Silla University, Korea**

Abstract

One of the major components of any pervasive system is its proactive behavior. Various models have been developed to provide system wide changes which would enable proactive behavior. A major drawback of these approaches is that they do not address the need to make use of existing applications without modifying the applications. To overcome this drawback, a middleware architecture called "Concord" is proposed. Concord is based on a simple model which consists of Lookup Server and Database. The rewards for this simple model are many. First, Concord uses the existing computing infrastructure. Second, Concord standardizes the interfaces for all services and platforms. Third, new services can be added dynamically without any need for reconfiguration. Finally, Concord consists of Database that can maintain and publish the active set of available resources. Thus Concord provides a solid system for integration of various entities to provide seamless connectivity and enable proactive behavior.

Key words : middleware, pervasive computing, ubiquitous, proactive behavior, Concord

1. Introduction

Pervasive computing is a trend towards an increasingly ubiquitous and connected computing devices communicating through interconnected networks[1] The major implications of this revolutionary technology can best be understood in terms of the realities and problems of using computer devices, and attempts to provide better solutions to tackle such problems and make life simpler and easier. Typical requirements of pervasive computing system are as follows[2, 3]. The system shall be dynamic at load-time. The system shall be able to discover and compose the services that are available in the physical environment. The computing devices shall be able to store program and user data in a local database. The computing devices shall be able to communicate with the network middleware. The communication mechanism shall allow the user to roam transparently between computing contexts. The computing devices shall support a wireless link. The wireless infrastructure shall support seamless roaming between short and long-range connections. Various models are currently being pursued to satisfy one or many of these requirements. Aura(Carnegie Mellon)[8], Oxygen(MIT) [7], Gaia[16] and PICO[17] are four such models. These models propose design changes in the way the system and computing devices are designed. A major drawback of these approaches is that they do not answer the need to make use of extant applications without changing the original designs of the applications to fit into the new environments.

In this paper a middleware architecture referred to as "Concord" is proposed to overcome the above drawback. Concord is based on a simple model which consists of Lookup Server and Database. The rewards for this simple model are as follows. Concord consists of Database which enables it to maintain and publish the active set of available resources. Concord standardizes the interfaces for all services and platforms. Concord uses the existing computing infrastructure. New services can be added dynamically without any need for reconfiguration. Section 2 provides an overview of the requirements of a good pervasive system and the various ongoing efforts in this field. Concord is proposed based on the outlined system requirements and the identified shortcomings of the various ongoing efforts. Section 3 discusses the architecture details of Concord. In section 4 a detailed explanation of Concord is provided with the help of a detailed system architecture diagram. Capabilities of Concord are demonstrated using real life scenarios, appropriate block diagrams and sequence diagrams. The details of inner-working of the design are included in order to provide a better understanding of how Concord enables proactive behavior in dormant environment.

Manuscript received May 27, 2005; revised July 21, 2005.

2. Requirements of a Pervasive Computing System

In this section we will discuss the technical requirements for implementing a successful pervasive computing model.

2.1 Device Requirements

a) The output mechanism should allow for visual, tactile, audio or data formats [3,4].

b) The input mechanism shall be either a touch screen or a speech recognition unit, or even a visual sensor [3,4].

c) There shall be sufficient storage space for the application data [3,4].

2.2 Network Communication

a) Any device that needs data or instruction in real time shall have the ability to communicate over the network[5].

b) The network shall consist of network devices and sensors that communicate over a wired or wireless infrastructure using different transport protocols [5].

c) The network communication shall be flexible to support different network configuration [5].

2.3 Communication Middleware

a) The infrastructure shall include communication middleware that connects the devices to various information, interface, and application servers [13, 14].

b) The communication middleware shall possess a discovery capability that can automatically connect the devices to new domains when the user enters a new context (logical or physical) [9].

c) The communication middleware shall possess capabilities for the devices to receive information relevant to the user's current context [10].

d) The communication middleware shall support query capability for the devices to look up information buffered by the infrastructure [12].

e) The communication middleware shall support publish or subscribe capability for the devices to receive information pushed from the infrastructure asynchronously [6].

f) The communication middleware shall support alert capability for the infrastructure to wake up the devices when establishing a communication channel [11].

2.4 Seamless Integration

The devices shall appear to users as universal translator allowing them to interact with intelligent devices.

3. Ongoing Pervasive Computing Efforts

3.1 Oxygen

Oxygen [7] is an approach to "invisible computing" being pursued by MIT. This approach has two primary aims:

a) Understand what turns an otherwise dormant environment into an empowered one.

b) Enable the user to shift much of the task burden onto the infrastructure.

Devices in Oxygen supply power for computation and communication, and work in much the same way that batteries and wall outlets supply power for electrical appliances. Both mobile and stationary devices are regarded as universal communication and computation appliances. They are also anonymous since they do not store configurations that are customized to any particular user[7].

The Oxygen project focuses on eight areas. The first is concerned with mobile service that relies on software to automatically detect and re-configure itself; the second and third technologies focus on embedded computing devices used to distribute computing nodes throughout the model, and network technology needed to allow embedded computing devices to interact and provide run time requirements. The final five technologies are all aimed at improving the user experience[7].

3.2 Aura

Aura[8] is described by Carnegie Mellon University as an architectural framework for user mobility in ubiquitous computing environments. The central architecture of Aura is designed to satisfy two competing goals for supporting mobile computational needs. The first is to maximize the use of available resources; i.e., to effectively exploit the increasingly pervasive computing and communication resources in modern environments. The second is to minimize the user distraction and attention[8]. Aura consists of the following components: Coda is a distributed file system. The Odyssey layer is for resource adaptation. Spectra is a remote execution service. Azure is a task layer that defines the interaction between layers that make tasks adaptable to a certain environment. AuraRT is a middleware that provides runtime support to the application. Prism is a task manager that interprets the human intention in any given situation[8].

Inherent to these pervasive computing efforts like Oxygen, Aura, etc. is a belief that traditional software will be treated as an application delivery mechanism: i.e., new software will support a more dynamic interface. However, this assumption turns out to be a major hurdle during application development, because these technologies work complementarily to the needs of any designer who may be developing an application to be used in conjunction with Oxygen. Overcoming this obstacle requires major changes in the way each application is designed and makes it difficult to use existing applications without requiring design changes.

4. System Architecture with Concord

Based on the above survey and requirement, proposed is a system model with Concord as middleware. The architecture of the proposed system is shown in Fig. 1.

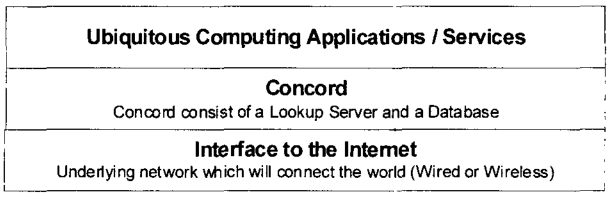

Ubiquitous Computing Applications / Services
Concord
Concord consist of a Lookup Server and a Database
Interface to the Internet
Underlying network which will connect the world (Wired or Wireless)

Fig. 1. System architecture of proposed system with Concord as middleware

4.1 Interface to the Internet

This layer provides communication capability over the network. This layer consists of communication, media technologies, and transport protocols.

4.2 Concord

Concord provides a distributed runtime' environment. Concord allows devices to interoperate and form spontaneous communities. To accomplish this, Concord supports discovery as well as storage capability. In this layer, devices export services that applications can use. Additionally, Concord supports publish and subscribe capability for devices to request information from infrastructure synchronously or receive information pushed from infrastructure asynchronously.

4.3 Ubiquitous Computing Applications / Services

The topmost layer of ubiquitous computing is an environment in which computing disappears into the background, weaving itself into everyday life. This layer enables a rich runtime environment. It has the ability to sense and interact with the computing environment: A characteristic defined as being "physical and real". Devices behave proactively depending on their surroundings.

5. Concord Architecture

This section gives a detailed description of Concord by covering the design aspects and providing an understanding of the different component modules and the reasons for their use.

5.1 Assumptions

As shown in Fig. 1, Concord resides between the application and the underlying Internet network structure with the following assumptions:

a) The network infrastructure is available for communication.
b) The interface to the network consists of devices and sensors along with transport protocols.
c) The transport technology may be Palm Pilots and Bluetooth or 802.11 type systems that provide short-range moderate bandwidth connections.
d) Widely deployed and easily accessible wireless Local Area Networks (LAN) and Wide Area Networks (WAN) will be available. ·

5.2 Model Description

Fig. 2 gives an overview of Concord, which consists functionally of two layers. Layer 1 consists of tasks. Tasks execute programs that form foundation to provide a transparent, ubiquitous computing and networking environment to an upper layer. The basic operation of the tasks is to create, start, and terminate a service. The tasks, like processes in traditional operating systems, are independent of each other, and local tasks interact by exchanging data. Layer 2 consists of Lookup Server and Database. Lookup Server maintains dynamic information about the available tasks, and Database represents persistent storage.

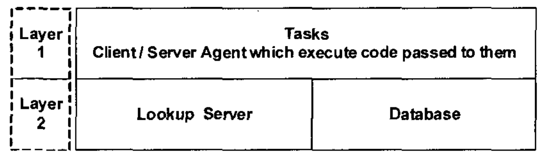


Fig. 2. Overview of Concord

5.3 Detailed Model Description

Fig. 3 details a layered architecture of Concord. The left side of the picture shows all the components of Lookup Server and the right side of the picture shows all the components of Database. Both Lookup Server and Database use components such as query and interface matching in different contexts, and the two do overlap.

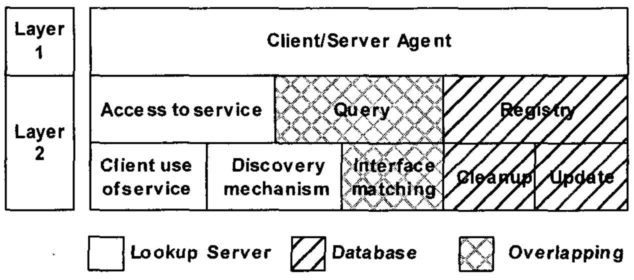


Fig. 3. Detailed architecture of Concord

As shown in Fig. 3, Layer 1 consists of tasks that can be any of the following:

a) Client Agent: Client Agent is a software component that runs in a device, such as PDAs, laptops, etc. It searches the network to find the services needed by the applications running in the device.
b) Server Agent: Server Agent runs on the devices, such as printers, plotters, scanners, etc., that provides the service. Server Agent is a software component that advertises the services provided by the device.

Layer 2 consists of Lookup Server and Database. Lookup Server and Database consists of the following components:

a) Discovery mechanism: Discovery mechanism defines how a client locates service discovery infrastructure.
b) Access to service: This topic addresses how a client, once it has located a service it needs, negotiates access to the service.
c) Client use of service: Once a client has located a service, and has successfully negotiated access to the service, it must send instructions in the form of an object ordered sequence of fields to the service. A field is defined as a basic component of the data structure hierarchy. For example, when a client requests for print service, it will send out a location, a type of file, a type of print and a number of copies to the print service.
d) Query: This is a mechanism by which Database is queried for the records.
e) Interface matching: Interface matching defines how a tem-

plate or an interface stored in a registry matched with the template provided by the client.

f) Registry: Registry is where the information about the available services is maintained. Data is stored as object ordered sequences of fields, and a field is defined as a basic component of the data structure hierarchy.

g) Update: This mechanism pertains to the protocols that Server Agent uses to update entries in Registry.

h) Cleanup: This module addresses how obsolete or incorrect information is purged from Registry.

5.4 Comparative Study of Concord vs. Other Systems

As illustrated in Table 1, the proposed system is centered on Concord. Concord has the following advantages compared to other models.

Table 1. Feature comparison of Aura vs. Oxygen vs. System with Concord

Feature	Aura	Oxygen	Concord
Does system support existing software?	No, design changes are needed to incorporate Aura's philosophy. For example use of AuraRT to provide distributed real-time object	No, design changes are needed to incorporate Oxygen's philosophy. For example use of second and third technology for embedded systems	Yes, Applications use standard interface to communicate with Concord.
Is system conducive to new software development	No. Developer is forced to follow Aura's philosophy.	No. Developer is forced to follow Oxygen's philosophy.	Yes. Concord uses standard interface to communicate with applications.
Light weight database	No, Aura provides developer with Coda a distributed files system	No, Oxygen defines a new technology for storing of data.	Yes, Database is lightweight in nature.
Proactive behavior	Yes	Yes	Yes
Scalability	Yes	Yes	Yes
Distributed application support	Yes	Yes	Yes
Accessibility	Yes	Yes	Yes
Invisibility	Yes	Yes	Yes
Fault tolerance	Yes	Yes	Yes

6. Case Study

What would it be like to live in a world with pervasive computing? To help convey the "look and feel" of such a world, real life scenarios are analyzed in this section. This section discusses the challenges in computer systems research posed by pervasive computing. We will now discuss two exemplary scenarios where Concord provides proactive behavior. These scenarios are chosen because they are typical in our day to day life and they embody many key ideas in pervasive computing.

6.1 Scenario 1

Mary is on a road in her car, driving through the city, she gets a call asking her to email an important time critical document. Client Agent running on Mary's cellular phone understands this and infers that Mary would like to use her wireless connection to e-mail this document. Unfortunately, she does not know where in the city she can access the Internet. Client Agent running in her system observes this. It consults a Location Server in the city and finds that wireless access is available at a nearby university library. A dialog box pops up on Mary's screen suggesting that she can go to the nearby university library, which is only three minutes away. Mary accepts the advice and drives to the library with the help of a map provided by Client Agent.

6.2 Scenario 2

Joe is in his office, frantically preparing for a meeting at which he will give a presentation in 15 minutes. It is time to leave, but he is not quite ready. He grabs his wireless handheld computer and walks out of the door. Client Agent infers where he is going from his calendar. It downloads the presentation and the demonstration software to the projection computer and warms up the projector. Joe has to provide a hard copy of his presentation for which the list of printers present in the locality is popped up; Joe chooses the nearest printer. He reaches the conference room and starts his presentation without any delay.

6.3 Missing Capabilities

These scenarios embody many key ideas in pervasive computing. Scenario 1 shows the importance of proactive behavior. Mary is able to complete her e-mail transmission only because Client Agent has capability to estimate how critical the whole process was and directed Mary to the right place where she could access the Internet. The scenario also shows the importance of combining knowledge from different layers of the system. Wireless connectivity is a low-level system phenomenon. Knowledge of university library is an application or user-level concept. Client Agent is able to obtain knowledge of wireless conditions at other places, and distance to these places because the computing environment provides these services. Scenario 2 illustrates the ability to move the execution state effortlessly across diverse platforms: from a desktop to a handheld machine and from the handheld to the projection computer. Self-tuning, or automatically adjusting behavior to fit circumstances, is shown by the ability to find printer. Scenario 2 embodies many instances of proactive behavior, by inferring that Joe is headed for the room across campus, warming up the projector, transferring the presentation and demonstration, anticipating that the he may request print service by combining this knowledge with the inferred past experience.

The biggest surprise in these scenarios is how simple and basic all the component technologies are. The hardware technologies (laptops, handhelds, wireless communication, software-controlled appliances, etc.) are all here today. The component software technologies have demonstrated location tracking, online database access, and so on. Then why do these scenarios seem like science fiction rather than reality? The answer lies in the fact that the whole system is much greater than the sum of its parts. In other words, the real research is in the seamless integration of component technologies into a single system. The difficult problems lie in architecture, component synthesis, and system-level engineering.

6.4 Solution

The following set of block diagrams and message sequence diagrams gives a better picture of how Concord enables seamless integration of various entities. An explanation of how each component in Concord provides a specific capability to achieve the final goal is explained later in the section.

6.4.1 Solution for Scenario 1

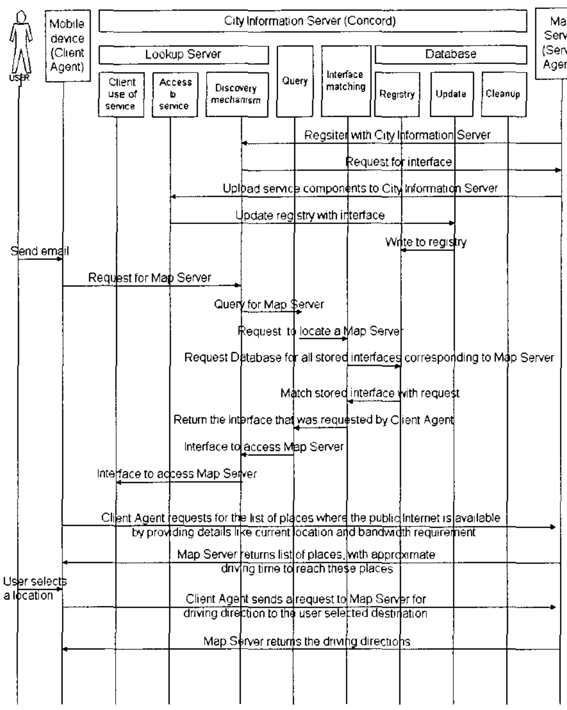
As shown in Fig. 4, Map Server connects to City Information Server asynchronously. Laptop computer represents a mobile user who will be in and out of the network as he moves around. City Information Server hosts Concord.

Layer 1	Tasks Client/ Server Agent which execute code passed to them	
Layer 2	Lookup Server	Database

Fig. 4. High level block diagram of Scenario 1

City Information Server maintains a list that specifies all available services on the network. Map Server maintains information of all the places and their specialty. Map Server provides driving directions to any user who requests directions from one place to another. Client Agent on the laptop requests City Information Server for a list of spots where the public internet is available. City Information Server returns an interface to Client Agent. This interface is used by Client Agent to communicate with Map Server. Calculating the latest position using the GPS, Client Agent fills in the interface with its position and its request for list of spots that offer access to the Internet. Map Server responds to Client Agent by providing a list of places where there is Internet availability. The user chooses the spot which he/she desires. Client Agent sends a request for directions to the spot selected by the user. Map Server provides directions to the spot.

As shown in Fig. 5, when Map Server wants to publish its capabilities, it first discovers one or many Lookup Server from the local or remote networks. Map Server then uploads its service components to City Information Server. The service proxy is stored in Registry. Client Agent can use this proxy to contact the original service and invoke methods on the service. Clients find a service by simple Interface matching.



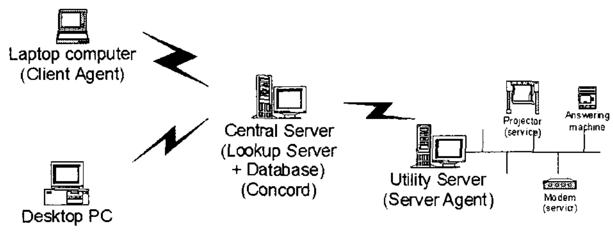
User
Mobile device (Client Agent)
City Information Server (Concord)
Lookup Server
Client use of service
Access to service
Discovery mechanism
Query
Interface matching
Database
Registry
Update
Cleanup
Map Server (Server Agent)
Regsiter with City Information Server
Request for interface
Upload service components to City Information Server
Update registry with interface
Write to registry
Send email
Request for Map Server
Query for Map Server
Request to locate a Map Server
Request Database for all stored interfaces corresponding to Map Server
Match stored interface with request
Return the interface that was requested by Client Agent
Interface to access Map Server
Interface to access Map Server
Client Agent requests for the list of places where the public Internet is available by providing details like current location and bandwidth requirement
Map Server returns list of places, with approximate driving time to reach these places
User selects a location
Client Agent sends a request to Map Server for driving direction to the user selected destination
Map Server returns the driving directions

Fig. 5. Inner working of different entities in Scenario 1

In Fig. 5, a user requests to send an email, Client Agent checks for the connectivity. When Client Agent finds that there is no connectivity. Client Agent requests City Information Server for the map service. City Information Server queries Database and provides an interface to Client Agent. Client Agent communicates with Map Server using the interface provided by City Information Server. Client Agent requests for the list of places where the public Internet is available by providing details like current location, bandwidth requirement, etc. Map Server returns list of places, with approximate driving time to reach these places. The user selects the location which suits him/her best. Upon selection, Client Agent sends a request to Map Server for driving direction to the user selected destination. Map Server returns the driving directions.

6.4.2 Solution for Scenario 2

This section explains the various aspects of the system used for the Scenario 2.



Laptop computer (Client Agent)
Desktop PC
Central Server (Lookup Server + Database) (Concord)
Utility Server (Server Agent)
Projector (service)
Answering machine
Modem (service)

Fig. 6. High level block diagram of Scenario 2

As shown in Fig. 6, Utility Server connects to Central Server asynchronously. The laptop computer represents a mobile user who will be in and out of the network as he moves around. Desktop computer represent a stationary user. Central Server hosts Concord. Utility Server provides services for projector, printer, scanner, etc.

Central Server maintains a list that specifies all available services on the network. Utility Server maintains information of all the devices and their location. Utility Server provides service to any user who requests service for printing, modem, projector, etc. When a mobile user schedules a presentation, Client Agent on the laptop requests an interface for the projector; Central Server returns an interface to Client Agent. Using this interface, Client Agent communicates with Utility Server, and Client Agent fills in the interface with its room number, requests for warming up the projector, and also requests for a list of all available printers near the conference room. Utility Server responds to Client Agent's request by providing a list of all the printers, and providing an interface which can control the projector. Client Agent sends a request to start up the projector and print the documents to any printer user desires.

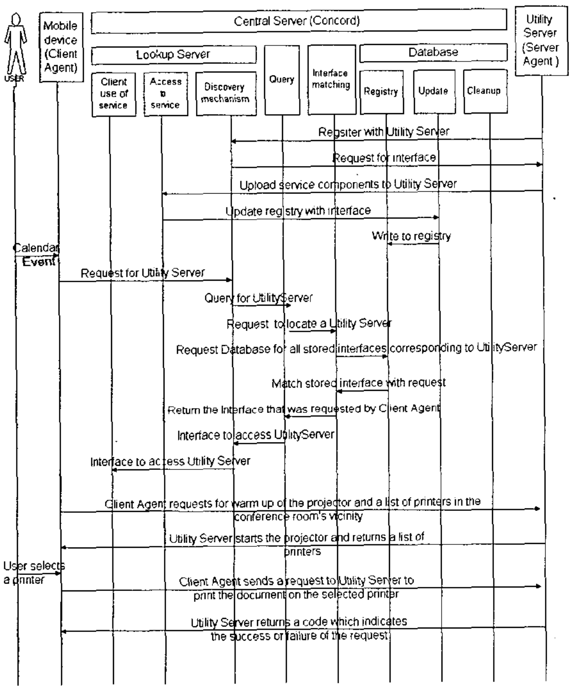

USER
Mobile device (Client Agent)
Central Server (Concord)
Lookup Server
Client use of service
Access to service
Discovery mechanism
Query
Interface matching
Database
Registry
Update
Cleanup
Utility Server (Server Agent)
Regsiter with Utility Server
Request for interface
Upload service components to Utility Server
Update registry with interface
Write to registry
Calendar Event
Request for Utility Server
Query for UtilityServer
Request to locate a Utility Server
Request Database for all stored interfaces corresponding to UtilityServer
Match stored interface with request
Return the Interface that was requested by Client Agent
Interface to access UtilityServer
Interface to access Utility Server
Client Agent requests for warm up of the projector and a list of printers in the conference room's vicinity
Utility Server starts the projector and returns a list of printers
User selects a printer
Client Agent sends a request to Utility Server to print the document on the selected printer
Utility Server returns a code which indicates the success or failure of the request

Fig. 7. Inner working of different entities in Scenario 2

As shown in Fig. 7, when Utility Server wants to provide any kind of service, it first discovers Lookup Server on the local or remote networks. Server Agent uploads a service proxy to Central Server. The service proxy is stored in Database. Client Agent uses this service proxy to contact Utility Server. Client Agent uses Interface matching to find a service. In Fig. 7, Client Agent is triggered when it finds that a presentation is scheduled in 15 minutes. Client Agent requests Central Server for projector and print service. Central Server queries Database and provides an interface to Client Agent. Using this interface, Client Agent communicates with Utility Server. Client Agent requests for warm up of the projector and a list of printers in the conference room's vicinity. Utility Server starts the projector and returns a list of printers, which satisfies the user requirements. The user selects a printer which suits him/her best. Upon selection, Client Agent sends a request to Utility Server to print the document on the selected printer. Utility Server returns a code which indicates the success or failure of the request.

7. Conclusion

The essence of any pervasive system is proactive behavior. For a system to enable proactive behavior, the system shall be dynamic at load-time, shall support discovery and compose services that are available in physical environment. Computing devices shall store software and user data in a local database. Computing devices shall be able to communicate with the network middleware, and the communication mechanism shall allow user to roam seamlessly between computing contexts. Finally, computing devices shall support a wireless link.

Various models such as Aura (Carnegie Mellon), Oxygen (MIT), Gaia, and PICO have been developed to provide system wide changes to enable proactive behavior. A major drawback of these approaches is that they do not answer the need to make use of extant applications without modifying the applications. The proposed middleware architecture, Concord, provides a platform for the integration of various entities to provide a seamless connectivity and enable proactive behavior.

Concord is a model bases on Lookup Server and Database. The rewards of this simple model are as follows: Concord standardizes the interfaces for all services and all platforms; Concord uses the existing computing infrastructure. New services can be added dynamically without any need for reconfiguration, and Concord consists of Database, which enables it to maintain and publish the active set of available resources. The capabilities of Concord are demonstrated using real life scenarios, appropriate block diagrams, and sequence diagrams. The details of the inner-working of the design are included in order to provide a better understanding of how Concord enables proactive behavior in dormant environment.

References

[1] M.Weiser, "Some Computer Science Issues in Ubiquitous Computing," *Communications of the ACM*, pp. 75-84, 1993.

[2] C. Huang, B. C. Ling and S. Ponnekanti, "Pervasive Computing: what is it good for?," *Proceedings of the ACM international workshop on Data engineering for wireless and mobile access*, pp. 84-91, 1999.

[3] M.Satyanarayanan, "Pervasive Computing: Vision and Challenges," *IEEE Personal Communications*, pp.10-17, 2001.

[4] G. Banavar, J. Beck, E. Gluzberg, J. Munson, J. Sussman and D. Zukowski, "Challenges: An Application Model for Pervasive Computing," *Proceedings of the 6th annual international conference on Mobile computing and*

networking, pp. 266-274, 2000.

[5] J. Flinn, D. Narayanan, and M. Satyanarayanan, "Self-Tuned Remote Execution for Pervasive Computing," *Proceedings of the 8th Workshop on Hot Topics in Operating Systems*, pp. 61-66, 2001.

[6] G. G. Richard III, "Service Advertisement and Discovery-Enabling Universal Device Cooperation," *IEEE Internet Computing*, pp. 18-26, 2000.

[7] http://Oxygen.lcs.mit.edu/Overview.html#Challenges.

[8] D.Garlan, S.Dan, S.Asim, and S. Peter, "Project Aura: Towards Distraction-Free Pervasive Computing," *IEEE Pervasive Computing, special issue on Integrated Pervasive Computing Environments*, vol. 1, no. 2, pp. 22-31, 2002.

[9] K.Takasugi, M.Nakamura, S.Tanaka, M.Kubota, "Seamless Service Platform for Following a User's Movement in a Dynamic Network Environment," *Proceedings of the First IEEE International Conference on Pervasive Computing and Communication*, pp. 71-78, 2003.

[10] C.Borceal, C.Intanagonwiwatl, A.Saxena, and L.Iftode, "Self-Routing in Pervasive Computing Environments using Smart Messages," *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications*, pp. 87-96, 2003.

[11] K.R¨omer, T.Schoch, F.Mattern, T.D¨ubendorfer, "Smart Identification Frameworks for Ubiquitous Computing Applications," Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, 2003, pp. 253-262.

[12] C. Bisdikian, J. Christensen, J. Davis, M. R. Ebling, G. Hunt, W. Jerome, H. Lei, S. Maes and D. Sow, "Enabling Location-Based Applications," *Proceedings of the first international workshop on Mobile commerce*, pp. 38-42, 2001.

[13] S. Cheng, D. Garlan, B. Schmerl, J. Sousa, B. Spitznagel, P. Steenkiste, and N. Hu, "Software Architecture-based Adaptation for Pervasive Systems," *International Conference on Architecture of Computing Systems: Trends in Network and Pervasive Computing*, pp. 67-82, 2002.

[14] Thayer and P. Steenkiste, "An Architecture for the Integration of Physical and Informational Spaces," *International Conference on Architecture of Computing Systems: Trends in Network and Pervasive Computing*, pp. 82-90, 2002.

[15] G.Judd and P.Steenkiste, "Providing Contextual Information to Pervasive Computing Applications," *IEEE International Conference on Pervasive Computing*, Vol 7,Issue 2, pp. 82-90, 2003.

[16] M.Roman,C.Hess, R.Cerqueira, A.Ranganathan, R.H.Campbell, K.Nahrstedt, "A middleware infrastructure for active spaces," *IEEE Pervasive Computing*, Vol 1 , Issue 4, pp. 74-83, 2002.

[17] M.Kumar, B.A.Shirazi, S.K.Das, B.Y.Sung, D.Levine, M.Singhal, "PICO: a middleware framework for pervasive computing," *IEEE Pervasive Computing*, Vol 2, Issue 3, pp. 72-79, 2003.

Sam Hsu
Current : Florida Atlantic University, Professor
Research Interests : Web Technologies, Web-based Distance Learning, Computer Internetworking, Practical & Educational Projects
E-mail : sam@cse.fau.edu

Mahesh Mutha
current : Motorola Inc., Plantation, Florida, USA
Email : emm053@motorola.com

A. S. Pandya
Current : Florida Atlantic University, Professor
Research Interests : Neural Networks, VLSI, Artificial Intelligence
E-mail : abhi@cse.fau.edu

YoungUhg Lho (Corresponding author)
He received the B.S., M.S. and Ph.D degrees in Dept. of Computer Science from Busan National University, Busan, Korea, in 1985, 1989, and 1998, respectively. From 1989~1996, he was with the Electronics and Telecommunications Research Institute(ETRI), Daejeon, Korea. Since 1996, he has been with the Dept. of Computer Education, Silla University, where he is now Associate Professor. His research interests include ubiquitous computing, embedded system, multimedia system, parallel and distributed system, intelligent system and computer education.

Phone : 051-309-5570
E-mail : yulho@silla.ac.kr