Context-Aware Application System for Music Playing Services*

Jae-Woo Chang and Yong-Ki Kim

Dept. of Computer Engineering
Center for Advanced Image and Information Technology
Chonbuk National University, Chonju, Chonbuk 561-756, South Korea
jwchang@chonbuk.ac.kr, ykkim@dblab.chonbuk.ac.kr

Abstract. Context-awareness is a technology to facilitate information acquisition and execution by supporting interoperability between users and devices based on users' context. In this paper, we design a middleware and a context server for dealing with context-aware application system in ubiquitous computing. The middleware plays an important role in recognizing a moving node with mobility as well as in executing an appropriate execution module according to context. In addition, the context server functions as a manager that efficiently stores context information, such as user's current status, physical environment, and resources of a computing system. Using them, we implement our application system which provides a music playing service based on context. It is shown to take below two seconds that our application system can detect a user's context and start playing music according to the context.

1 Introduction

Mark Wieser at Xerox Palo Alto Research Center described ubiquitous computing as being about interconnected hardware and software that are so ubiquitous that no one notices their presence [1]. An effective software infrastructure for running ubiquitous applications must be capable of finding, adapting, and delivering the appropriate applications to the user's computing environment based on the user's context [2]. Thus, context-aware application systems determine which user tasks are most relevant to a user in a particular context. They may be determined based on history, preferences, or other knowledge of the user's behavior, as well as the environmental conditions. The context-awareness can facilitate information acquisition and execution by supporting interoperability between users and devices based on users' context, in a variety of applications including location-based services and Telematics. In this paper, we design middleware and context server components for context-aware application systems. The middleware plays an important role in recognizing a moving node with mobility as well as in executing an appropriate execution module according to context. In addition, the context server functions as a manager that efficiently stores

* This work is financially supported by the Ministry of Education and Human Resources Development (MOE), the Ministry of Commerce, Industry and Energy (MOCIE) and the Ministry of Labor (MOLAB) though the fostering project of the Lab of Excellency.

B. Gabrys, R.J. Howlett, and L.C. Jain (Eds.): KES 2006, Part III, LNAI 4253, pp. 76 – 83, 2006.

context information, such as user's current status, physical environment and resources of a computing system. Using them, we implement our application system which provides a music playing service based on context.

2 Related Work

Context-aware application systems determine which user tasks are most relevant to a user in a particular context. They may be determined based on history, on preferences, or other knowledge of the user's behavior, as well as the environmental conditions. In this section, we discuss the typical context-aware application systems. First, INRIA in France [3] proposed a general infrastructure based on contextual objects to design adaptive distributed information systems in order to keep the level of the delivered service despite environmental variations. The contextual objects (COs) were mainly motivated by the inadequacy of current paradigms for context-aware systems. The use of COs does not complicate a lot of development of an application, which may be developed as a collection of COs. The value of a particular object used in a context-dependent application is automatically updated by the adaptive framework, independently from the application. Secondly, AT&T Laboratories Cambridge in U.K [4] presented a platform for context-aware computing which enables applications to follow mobile users as they move around a building. The platform is particularly suitable for richly equipped, networked environments. Users are required to carry a small sensor tag, which identifies them to the system and locates them accurately in three dimensions. Thirdly, Arizona State Univ. [5] presented Reconfigurable Context-Sensitive Middleware (RCSM), which made use of the contextual data of a device and its surrounding environment to initiate and manage ad hoc communication with other devices. The RCSM provided core middleware services by using dedicated reconfigurable FPGA (Field Programmable Gate Arrays), a context-based reflection and adaptation triggering mechanism, and an object request broker that are context-sensitive and invokes remote objects based on contextual and environmental factors, thereby facilitating autonomous exchange of information. Finally, Lancaster Univ. in U.K [6] presented a comprehensive description of the GUIDE project which has been developed to provide city visitors with a hand-held context-aware tourist guide. The development of GUIDE has involved: capturing a real set of application requirements, investigating the properties of a cell-based wireless communications technology in a built-up environment and deploying a network based on this technology around the city, designing and populating an information model to represent attractions and key buildings within the city, prototyping the development of a distributed application running across portable GUIDE units and stationary cell-servers.

3 Middleware and Context Server for Context-Awareness

Context is any information that can be used to characterize the situation of any entity [7]. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves. We design an overall architecture of context-adaptive computing system for supporting various context-aware application services by combining the advantage of the INRIA

work [3] with that of the AT&T work [4]. The system is divided into three components, context server, fixed node (middleware), and moving node (client). First, the context server functions as inserting remote object and context information into object and context database, respectively, and retrieving them from the databases. Secondly, a fixed node functions as a middleware, which can find, insert, and execute a remote object for context awareness. Finally, a moving object communicates with a fixed node and executes a predefined built-in program according to the context information acquired from a middleware. The context server communicates with a fixed node by using a network based on TCP/IP, while a moving object communicates with a fixed node using Bluetooth wireless communication [8]. The proposed context-aware computing system has a couple of powerful features. First, our middleware can define context objects describing context information as well as can keep track of a user's current location. Secondly, our context server can store context objects and their values depending on a variety of contexts as well as can manage users' current locations being acquired from a set of fixed nodes by using spatial indexing. Finally, our client can provide users with adaptive application services based on the context objects. Figure 1 shows the overall architecture for supporting various context-aware application services.

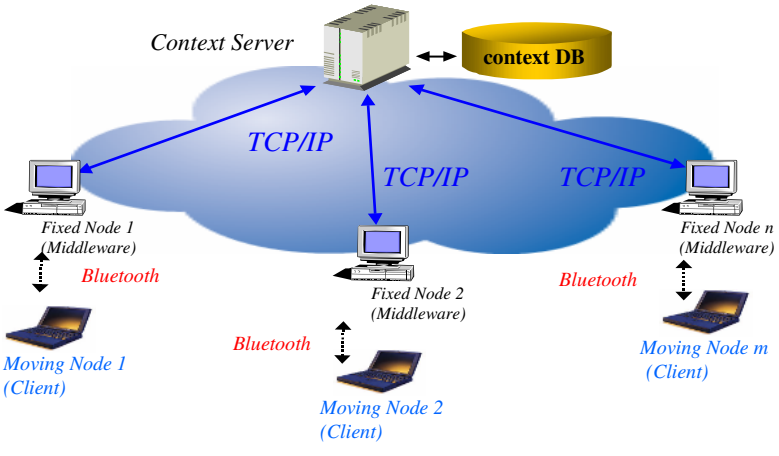

Fig. 1. Overall architecture for supporting context-aware application services

Our middleware for context-aware application services consists of three layers, such as detection/monitoring layer, context-awareness layer, and application layer. First, the detection/monitoring layer serves to monitor the locations of remote objects, network status, and computing resources, i.e., CPU usage, memory usage, bandwidth, and event information related with devices. Secondly, the context-awareness layer serves as a middleware which is an essential part for handling context-aware application services. It can be divided into five managers, such as script processor, remote object manager, context manager, context selection manager, communication proxy manager. The script processor analyzes the content of context-aware definition script and executes its specified action. The remote object manager manages a data structure

for all the context objects used in application programs. The context manager manages context and environmental information including user preference, user location, etc. The context selection manager chooses the most appropriate context information under the current situation. The communication proxy manager serves to communicate with the context server and temporarily reserve data for retransmission in case of failure. Finally, the application layer provides functions to develop various context-aware applications using the application programming interface (API) of the middleware while it is executed independently of the middleware.

The procedure to execute proper services based on context using our middleware has three steps. First, a moving node broadcasts a connection request signal so as to connect with a fixed node by using Bluetooth. The fixed node covering an interesting area analyzes the signal and accepts the connection with the corresponding fixed node. When the connection between them is established, the moving node delivers its own information to the fixed node. Secondly, when remote objects are found, the fixed node delivers their information to the context server through a network using TCP/IP. The sever stores into the context database the information delivered from the fixed node. Finally, the server searches the fixed node's context which is the most suitable with current situation from the context database. The fixed node executes predefined programs based on the context information and notifies a moving node of services being executed. Then, the moving node executes the service being requested by the fixed node. Because a moving node maintains all the addresses of fixed nodes and makes a connection to a fixed node, a moving node communicates with the fixed node periodically once a connection between them is established, and determines whether or not the connection between them should hold.

Because a context server is required to store and retrieve remote objects and context information extracted from them, we design a context server to efficiently manage both the remote object and the context information using a commercial DBMS. The context server analyzes and executes the content of packets delivered from the middleware. It is divided into four managers, such as communication manager (CM), packet analysis manager (PAM), context/object manager (COM), and SQL query manager(SQM). The CM serves as communicate between the server and the middleware. The CM delivers to PAM packets transferred from the middleware so as to parse them, and it delivers to the middleware result packets made from the server. The PAM parses the packets from the CM and determines what action wants to be done currently. Based on the parsing, the PAM calls the most proper functions in the COM. The COM translates into SQL statements the content delivered from the PAM and delivers the SQL statements to SQM to execute them. The application programming interface (API) for the COM is ContextDefine, ContextDestroy, ContextInsertTuple, ContextDeleteTuple, ContextSearch, ContextSearchTuples, and ContextCustom. The SQM executes the SQL statements from the COM by using the DBMS and delivers the result to the middleware via the CM. The API for the SQM is sql-Reader and sql-NonQuery.

4 Context-Aware Music Playing Application System

Using the middleware and the context server, we develop our context-aware music playing application system under Redhat Linux 7.3(kernel version 2.4.20) with 866

MHz Pentium-III CPU and 64 MB main memory. We make use of GCC 2.95.4 as a compiler and affix 2.0.2 as a Bluetooth device driver. The Bluetooth device follows the specification of Version1.1/Class1 and makes a connection to PCs using USB interfaces [9]. We also use MySQL DBMS as a commercial DBMS because we can reduce the developing time, compared with using a storage system, and we can increase the reliability of developed systems. In our music playing application system, when a user belonging to a moving node approaches to a fixed node, the fixed node plays a music with the user's preference according to the user's location. In general, each user has a list of his music with his(her) preference and moreover has different lists of popular music depending on time, i.e., morning time, noon time, and night time. Thus, when a user, which is listing to his popular music in the area of the fixed node 1, moves to the area of the fixed node 2 (Figure 1), the music stops playing in the area of the fixed node 1 while it starts playing in the area of the fixed node 2. The fixed node differentiates a user from another user and plays his(her) preferred music depending on the current time by considering the time when a user enters into the area of the fixed node. The record of a database in context server for our music playing application has six attributes, such as User_ID, User_Name, Location, Music_M, Music_A, and Music_E. The User_ID serves as a primary key to identify a user uniquely. The User_Name means a user name and the Location means a user's current location which can be changed whenever a fixed node finds the location of a moving object. Finally the Music_M, the Music_A, and the Music_E represent his(her) popular music file in the morning time, the noon time, and the night time, respectively.

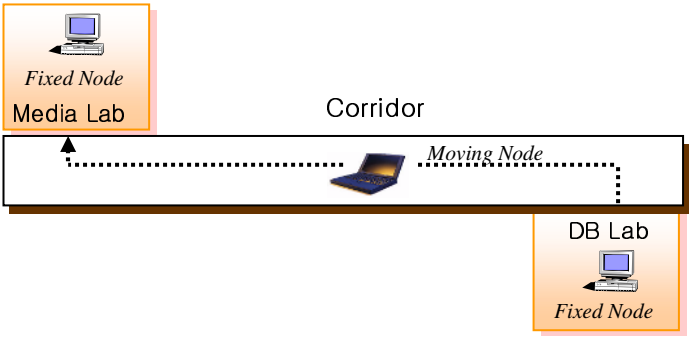

Fig. 2. Testing environment for our application system

To determine whether or not our application system works well, we test it by adopting the scenario used in Cricket [10], one of the MIT Oxygen project. We test its execution in the following three cases; the first is when a user covered by a moving node approaches to a fixed node or move apart from it, the second is when two different users approaches to a fixed node, and final case is when a user approaches to a fixed node at different times. Among them, because the fist case is the most general case, we will explain the first case in more detail. For our testing environment, we locate two fixed nodes in Database laboratory (DB Lab) and Media communication laboratory (Media Lab) of Chonbuk National University, as shown in Figure 2, where

the fixed node can detect a moving node by using Bluetooth wireless communication. There is a corridor between DB Lab and Media Lab and its distance is about 60 meter. We test its execution in case when a user having a moving node moves from DB Lab to Media Lab or in a reverse direction.

Figure 3 shows testing in case when a user having a moving node is approaching to a fixed node. First, the fixed node receives a user name from the moving node as the moving node is approaching to it (①). Secondly, we determine whether the information of the user has already been stored into a server or not. If does, we search the music file belonging to the user in a given time and downloads the music file from the server (②). Finally, we play the downloaded music file by using a MP3 music player. On the contrary, testing in case when a user having a moving node is moving apart from to a fixed node is so little different. First, when the middleware detect that a user is too far from the fixed node to communicate with it, we output an error message and stop the process playing the music. Finally, we remove the music player process. In a short, when a user is approaching to a fixed node, the music belonging to the user is playing while when a user is moving apart from the fixed node, the music stops playing.

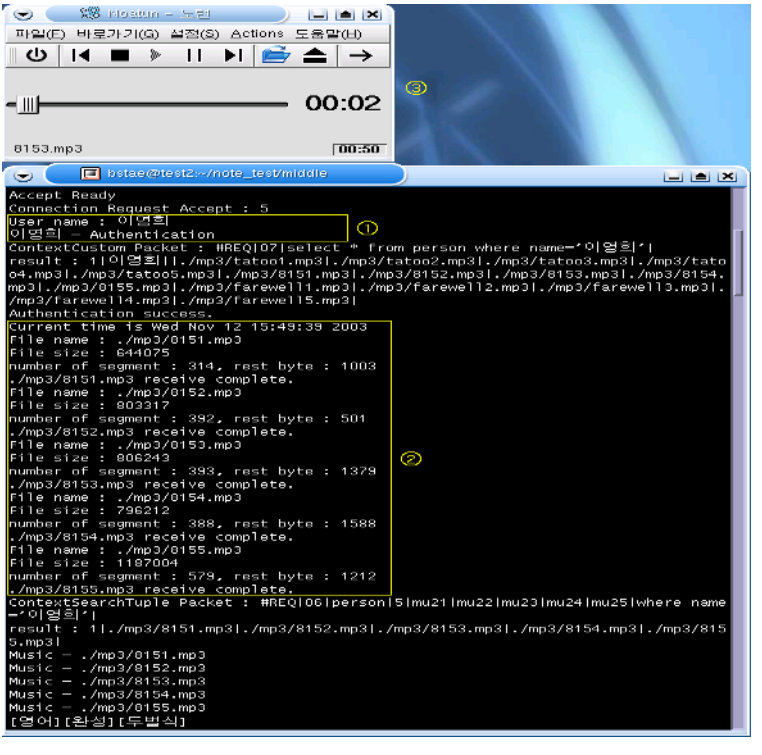

Fig. 3. Testing in case when a user is approaching to a fixed node

For the performance analysis of our application system, we measure the average times by adopting a boundary detection of beacons used in Cricket [10]. First, as a moving node is approaching to a fixed node, it takes 1.34 second to make a connection

between them. The time means the one to detect the connection of a moving node by middleware when a moving node enters into the communication boundary of a fixed node. The time mainly depends on the specification of Bluetooth wireless communication. Secondly, it takes 0.5 second to start a corresponding service after making the connection between them. The time means the one to search the profile of the corresponding user and to call the module playing music by the middleware. Here the searching time for a user's profile is dependant on the packet transfer time of TCP/IP and on the DBMS performance of the context server. The calling time for the music playing module means the one for loading it by an operating system (OS), which is affected by the available memory of the OS kernel and the speed of a hard disk. Finally, as a moving node is moving apart from a fixed node, it takes 1.45 second to make a disconnection between them. The time means the one to detect the disconnection of a moving node by the middleware. The time is relatively long due to the kernel's release of socket resources because the kernel tries to communicate with the moving node even though the moving node goes out of the communication boundary of a fixed node. When the kernel is connected or disconnected with the moving object, it can be considered very reasonable that the middleware sets the time limit to be two seconds. Thus, if it takes over two seconds for the middleware to make a connection with a moving node and detect context from it, a user may consider the situation as a fault.

5 Conclusions and Future Work

In this paper, we designed both a middleware and a context server for dealing with context-aware application system in ubiquitous computing. The designed middleware played an important role in recognizing a moving node with mobility by using a Bluetooth wireless communication as well as in executing an appropriate execution module according to the context acquired from a context server. The designed context server functioned as a manager that efficiently stores into the database server context information, such as user's current status, physical environment, and resources of a computing system. Using the middleware and the context server, we implemented our application system which provides a music playing service based on context. It was shown to take below 2 seconds that our application system could detect a user's context and start playing music according to the context. As future works, it is required to study on an inference system to acquire new context information from the existing context information.

References

1. M. Weiser, "Some Computer Science Issues in Ubiquitous Computing", Commnunication of the ACM, Vol 36(7), pp. 75-84, 1993.
2. G. Banavar, A. Bernstein, "Issues and challenges in ubiquitous computing: Software infrastructure and design challenges for ubiquitous computing applications", Communication of ACM, Vol 45(12), pp. 92-96, 2002.
3. P. Couderc, A. M. Kermarrec, "Improving Level of Service for Mobile Users Using Context-Awareness", Proc. of 18th IEEE Symposium on Reliable Distributed Systems, pp. 24-33, 1999.

4. A. Harter, A. Hopper, P. Steggles, A. Ward, P. Webster, "The anatomy of a Context-aware application", Wireless Networks Vol. 8, Issue 2/3, pp. 187-197, 2002.
5. S. S. Yau and F. Karim, "Context-sensitive Middleware for Real-time Software in Ubiquitous Computing Environments", Proc. of 4th IEEE Symposium on Object-oriented Real-time Distributed Computing, pp.163-170, 2001.
6. K. Cheverst, N. Davies, K. Mitchell, A. Friday, "Experiences of developing and deploying a context-aware tourist guide: the GUIDE project", Proceedings of the sixth annual international conference on Mobile computing and networking, pp. 20-31, 2000.
7. A. K. Dey, "Understanding and Using Context", Personal and Ubiquitous Computing Journal, Vol. 5, No. 1, pp. 4-7, 2001.
8. Bluetooth Version 1.1 Profile, http://www.bluetooth.com.
9. Affix: Bluetooth Protocol Stack for Linux, http://affix.sourceforge.net.
10. N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The Cricket Locaion Support System", 6th ACM/IEEE Int'l Conf. on Mobile Computing and Networking(MOBICOM), pp. 32-43, 2000.