Design and Implementation of Middleware and Context Server for Context-Awareness*

Jae-Woo Chang and Yong-Ki Kim

Research Center of Industrial Technology
Dept. of Computer Engineering, 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 and implement a middleware and a context server for dealing with context-aware applications in pervasive computing. The middleware plays an important role in recognizing a moving node with mobility by using a Bluetooth wireless communication technology as well as in executing an appropriate execution module according to the context acquired from a context server. In addition, the context server functions 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. To verify the usefulness of the middleware and the context server, we finally develop our context-aware application system which provides users with a music playing service in pervasive computing environment.

1 Introduction

In traditional computing environments, users actively choose to interact with computers. On the contrary, pervasive computing applications are embedded in the users' physical environments and integrate seamlessly with their everyday tasks [1]. Mark Wieser at Xerox Palo Alto Research Center identified the goal of future computing to be invisible computing [2]. An effective software infrastructure for running pervasive computing applications must be capable of finding, adapting, and delivering the appropriate applications to the user's computing environment based on the user's context. 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. Once the user has selected a task from the list of relevant tasks, an application may have to move seamlessly from one device to another and from one environment to another based on the user's activity. The context-awareness is one of the most important technologies in pervasive computing, which facilitate information acquisition and execution by supporting interoperability between users and devices based on users' context.

* 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.

M. Gerndt and D. Kranzlmüller (Eds.): HPCC 2006, LNCS 4208, pp. 487 – 494, 2006.

In this paper, we design and implement middleware and context server components for dealing with context-aware applications in pervasive computing. The middleware plays an important role in recognizing a moving node with mobility by using a Bluetooth wireless communication technology as well as in executing an appropriate execution module according to the context acquired from a context server. In addition, the context server functions as a manager that efficiently stores into database server con-text information, such as user's current status, physical environment, and resources of a computing system. In order to verify the usefulness of the middleware and the con-text server, we develop our context-aware application system which provides a music playing service in pervasive computing environment. The remainder of this paper is organized as follows. The next section discusses related work. In section 3, we de-scribe the overall architecture for context-aware application services. In section 4 and 5, we describe the design of our middleware and our context server for context-awareness. In section 6, we present the development of our context-aware application system using them. Finally, we draw our conclusions in section 7.

2 Related Work

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. They also presented a general framework for context-aware systems, which provides application developers with an architecture to design and implement adaptive systems and supports a wide variety of adaptations. 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. Finally, 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 Field Programmable Gate Arrays (FPGA), 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.

3 Overall Architecture for Context-Aware Application Services

Context is any information that can be used to characterize the situation of any entity [6]. 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. In this section, we propose an overall architecture of context-adaptive system for supporting various context-aware application services, which is divided into three com-

ponents, context server, middleware (fixed node), and moving node (client). First, the context server serves to insert remote objects into an object database and context information into a context database, as well as to retrieve them from the both databases. Secondly, a fixed node functions as a middleware to find, insert, and execute a remote object for context awareness. Finally, a moving object serves to execute a predefined built-in program according to the context information acquired from the middleware. Figure 1 shows the overall architecture for supporting various context-aware application services.

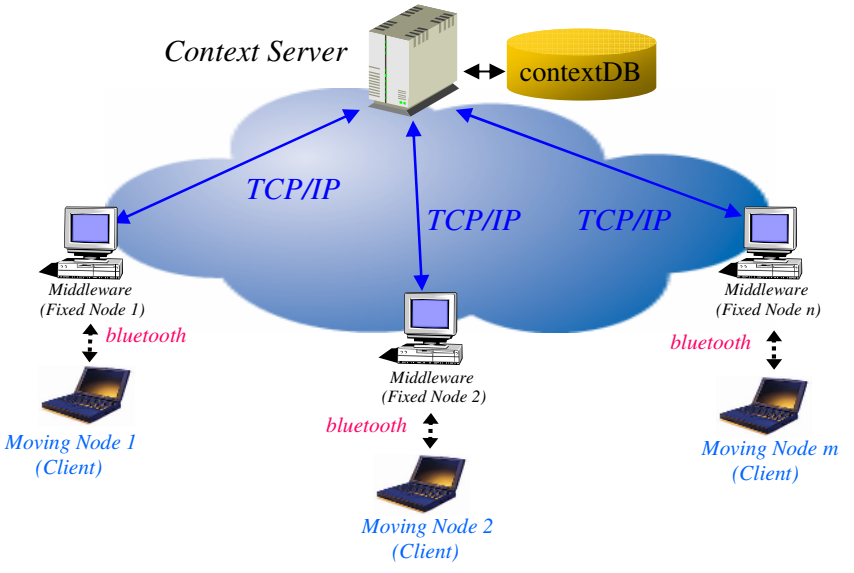

Fig. 1. Overall architecture for supporting context-aware application services

Because our architecture combines the advantage of the INRIA work with that of the AT&T work, it 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 mange 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. Meanwhile, the context server communicates with a middleware by using a network based on TCP/IP, while a moving object communicates with a middleware using Bluetooth wireless communication [7].

4 Middleware for Context-Awareness

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 including Bluetooth. Secondly, the context-awareness layer functions 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 actions. The remote object manager man-ages a data structure for all the context objects used in application programs. The context manager manages context and environmental information including user preference and user location. 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 to temporarily reserve data for retransmission in case of failure. Finally, being executed independently of the middleware, the application layer provides users with a set of functions to develop various context-aware applications by using the application programming interface (API) of the middleware. Figure 2 shows the three-layered structure of the middleware.

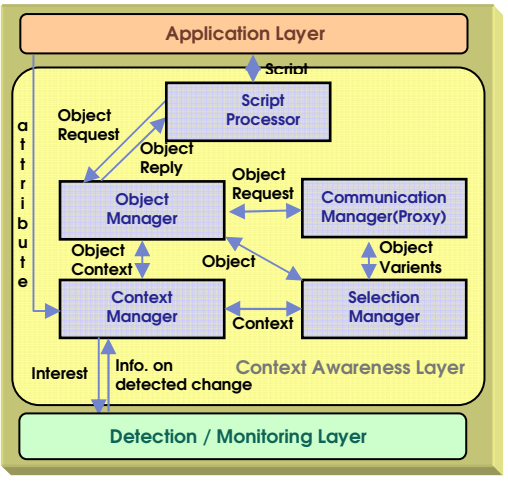

Fig. 2. Three-layered structure of the middleware

5 Context Server for Context-Awareness

For context-awareness, context server is required to store and retrieve remote objects and context information extracted from them. We design a context server which can efficiently manage both the remote object and the context information using a commercial DBMS called MySQL. This is because we can reduce the developing time, compared with using a storage system, and we can increase the reliability of the developed system. The designed context server analyzes and executes the content of packets delivered from the middleware. That is, the server determines whether the packet's content is contexts or context objects and stores them into the corresponding database. Figure 3 shows the structure of context server which is divided into four managers, such as communication manager (CM), packet analysis manager (PAM), context/object manager (COM), and mySQL query manager (SQM).

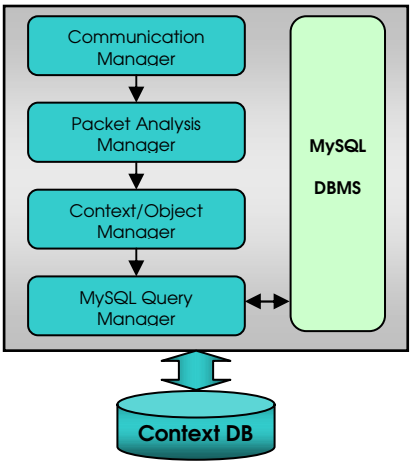

Fig. 3. Structure of context server

The CM serves to communicate between the context server and a middleware. The CM delivers to PAM the packets transferred from the middleware as well as to the middleware the result packets made from the server. The CM includes both file receiving module and TCP/IP socket module. The file receiving module is dependant on the TCP/IP socket module because it communicates with the middleware using TCP/IP socket. When the server is ready to communicate, it receives packets from the middleware. 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, i.e., context APIs, 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 context APIs (application programming interfaces) for the COM is ContextDefine, ContextDestroy, ContextInsertTuple, ContextDeleteTuple, ContextSearch, ContextSearchTuples, and ContextCustom. The SQM executes the SQL statements from the COM using the MySQL DBMS and delivers the result to the middleware via the CM. The SQL includes the mySQL API module being implemented by using mysql C libraries.

6 Development of Context-Aware Application System

In this section, we first develop both our middleware and our context server which are designed for context awareness in the previous sections. For this, we implement them using GCC compiler 2.95.4 under Redhat Linux 7.3 (kernel version 2.3.20) with 1.7 GHz Pentium-IV CPU and 512 MB main memory. In order to show the efficiency of both our middleware and our context server implemented, we also develop a con-text-aware application system using them. The context-aware application system servers to provide users with a music playing service in pervasive computing environment. In the system, when a user belonging to a moving node approaches to a fixed node, the fixed node starts playing the user's music with his (her) preference according to his

location. In general, each user has a list of his (her) music with his preference and even a user can have a different list of his (her) popular music depending on time, i.e., morning time, noon time, and night time. In the context server, a user record for the music playing application service 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 middleware finds the location of a moving object. Finally the Music_M, the Music_A, and the Music_E represent his (her) preferred music file in the morning time, the noon time, and the night time, respectively. The context server manages a list of music files for a user, processes queries given from a fixed node, and delivers the corresponding music file to the fixed node by its request.

We develop our context-aware application system providing a music playing service by using affix 2.0.2 as a Bluetooth device driver protocol and by using GCC 2.95.4 an a compiler, under Redhat Linux 7.3 (kernel version 2.4.20) with 866 MHz Pentium-III CPU and 64 MB main memory. In addition, the Bluetooth device follows the specification of Version1.1/Class1 and makes a connection to PCs using USB interfaces [8]. To determine whether or not our context-aware application system implemented works well, we test it by adopting a scenario used in Cricket [9], one of the MIT Oxygen project. For this, we test the execution of our context-aware application system in the following three cases; the first case when a user covered by a moving node approaches to a fixed node or move apart from it, the second case when two different users approaches to a fixed node, and the final case when a user approaches to a fixed node at different times. Among them, because the first case is the most general one, we will explain it in more detail. For our testing environment, we locate two fixed nodes in the database laboratory (DB Lab) and the media communication laboratory (Media Lab) of Chonbuk National University, respectively, where their middleware 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 the execution of our context-aware application system in a case when a user having a moving node moves from DB Lab to Media Lab or in a reverse direction. Figure 4 shows a testing case when a user having a moving node approaches 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, the fixed node determines whether or not the information of the user has already been stored into a server. If it does, the context server searches the music file belonging to the user in a current time and downloads the music file from the database (②). In case when the fixed node detects that a user is too far to communicate with it, the fixed node stops the process to play music and it re-moves the music playing process.

To analyze the performance of our context-aware application system, we measure an average time by adopting a boundary detection of beacons used in Cricket [9]. First, as a moving node is approaching to a fixed node, it takes 1.34 second for the fixed node to make a connection with the moving node. It means the time for the fixed node to detect the presence of a moving node. Secondly, it takes 0.5 second for the fixed node to start music playing service after making the connection between them. Finally, as a moving node is moving apart from a fixed node, it takes 1.45

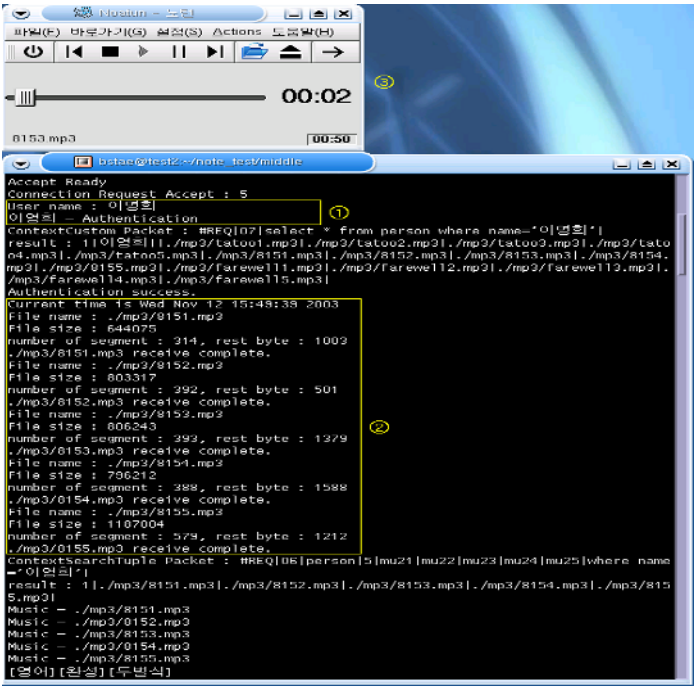

Fig. 4. Testing case when a user is approaches to a fixed node

second for the fixed node to make a disconnection to the moving node. It means the time for the fixed node to detect the absence of the moving node. The time is relatively long because the kernel tries to communicate with the moving node even though the moving node is beyond the communication boundary of the fixed node. Therefore, it is very reasonable for the fixed node to set the time limit to two seconds. If it takes long time for a fixed node to establish a connection to a moving node and to detect a context from it, a user may consider the situation as a fault. Because the detection time for a context is less than two seconds, the context-aware application program is reasonable for the music playing application service.

7 Conclusions and Future Work

In this paper, we designed and implemented both our middleware and our context server for supporting a context-aware application system. The middleware played an important role in recognizing a moving node with mobility by using Bluetooth wireless communication as well as in executing an appropriate execution module according to the context acquired from a context server. In addition, the context server functions as a manager that efficiently stores into database server context information, such as user's current status, physical environment. To verify the usefulness of both the middleware and context server implemented, we developed our context-aware application system which provided users with a music playing service in pervasive computing environment. We tested it by adopting a scenario used in Cricket. It was

shown that it took about 1.5 seconds for our context-aware application system to make a connection (or disconnection) between a fixed node and a moving node, thus being considered reasonable for our music playing application service. As future work, it is required to study on an inference engine to acquire new context information from the existing one.

References

1. K. Raatikainen, H. B. Christensen, and T. Nakajima, "Application Requirements for Middleware for Mobile and Pervasive systems", Mobile Computing and Communications Review, pp. 16-24, Vol. 6, No. 4.
2. M. Weiser, "The Computer for the Twenty-First Century", Scientific American, pp. 94-104, Sept. 1991.
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, and A. Feiday, "Experiences of Developing and Deploying a Context-Aware Tourist Guide: The GUIDE Project", Proc. of 6th Int'l Conference on Mobile Computing and Networking, 2001.
7. Bluetooth Version 1.1 Profile, http://www.bluetooth.com.
8. Affix: Bluetooth Protocol Stack for Linux, http://affix.sourceforge.net.
9. N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The Cricket Location Support System", 6th ACM/IEEE Int'l Conf. on Mobile Computing and Networking(MOBICOM), pp. 32-43, 2000.