Proceedings of the 4th WSEAS Int. Conf. on Information Security, Communications and Computers, Tenerife, Spain, December 16-18, 2005 (pp139-142)

Terminal application middleware for the interoperability among telematics services

Moon-Soo Lee, Jong-Woo Choi, Myung-Jin Lee, Kwon Oh-Cheon
Telematics S/W Platform Team, Telematics•USN Research Division
Electronics and Telecommunications Research Institute(ETRI)
161 Gajeong-dong, Yuseong-gu, Daejeon
KOREA

Abstract: Telematics offers automobile drivers with various and useful information services from telematics service provider (TSP) centers through a terminal in vehicle which is installing wireless communication devices such as CDMA, GSM, DSRC, etc. However, a vehicle environment, which telematics service is used, has poor surroundings in the large temperature variation and noise. Because of these conditions, Telematics terminals were developed with relative importance in stability itself more than the interoperability for the internal software circumstances. Therefore, software in the terminals has a lot of dependencies on its hardware conditions and the interoperability among contents can not be guaranteed any more.
This paper introduces an application middleware for telematics terminals which was developed as a kind of telematics technology development project. The middleware based on open standard of JAVA technology can make operation among contents over itself. Because of providing standard interfaces of the middleware, it can make an easy to develop telematics services.

Key-Words: - Telematics, Terminal, Middleware, Platform, TSP, Vehicle

1 Introduction

Expansions of road can make interchange among areas which is isolated at the past. As a lot of information is shared through inter-communication of those areas, it is gradually progressed as a high information society. Though vehicle is a spot which many people are using and killing their times, it was relatively isolated place far from the Internet called with high information road. However, with the help of the advance of embedded systems and mobile communications such as CDMA, the usability of vehicle is increased in the mobile environment gradually. In addition, it is required to the high quality of telematics services.
Recent telematics terminal is designed to have robustness against high temperature and noises in car. It is built up an embedded system with 6-inch screen size and similar type of AutoPC. But services and contents of terminal were inevitable to be redeveloped even though they were made in the same company. Because each contents in the terminal is optimized in its hardware. They also have to download their dedicated contents from telematics service center (TSP).
OSGi (Open Service Gateway Initiative) offers a basis framework which has core functions of registration, discovery, and execution of services to support various devices in home or vehicle. OSGi ensures the interoperability between services overt its framework. And it puts up an environment which can be deployed and executed service-oriented applications based on component model.
In this paper introduces a telematics application middleware, which supports to reduce the term and cost in the development of telematics contents because of increasing their reusability and guaranteeing their compatibility. As it adopts an open standard technology of JAVA language, the contents are able to applicable independent to the hardware and operating systems of the terminal.
The rest of this paper is structured as follows: Section 2 provides an overall architecture of terminal S/W platform and introduces OSGi framework and AMI-C specification, Section 3 describes to the terminal application middleware based on AMI-C. Finally, Section 4 concludes the paper.

2 Terminal S/W platform architecture

Embedded software implemented in the initial telematics terminal was developed and optimized in low performance hardware limited resources. As telematics terminals as well as services are various, the

interoperability between the services is needed gradually.
Fig. 1 shows the overall architecture of telematics terminal platform. The basis structure of the platform has embedded operation system such as WinCE .NET, embedded linux over the hardware of the terminal and JAVA virtual machine exists on it. Middleware represented in this paper is operating over the JVM following the J2ME specification. It is comprised of OSGi framework originally targeted in digital home gateway and AMI-C specification.
A standard telematics API provided terminal S/W platform contained the middleware helps service developers to use a uniform program interfaces for the devices in the vehicles. Applications to be developed in this way can be made to the emergency service, remote automobile monitoring service, PIMS, etc. The services will be deployed and executed over the telematics terminal S/W platform.
In this chapter will describe the OSGi and AMI-C specifications that are the core of the terminal S/W platform.

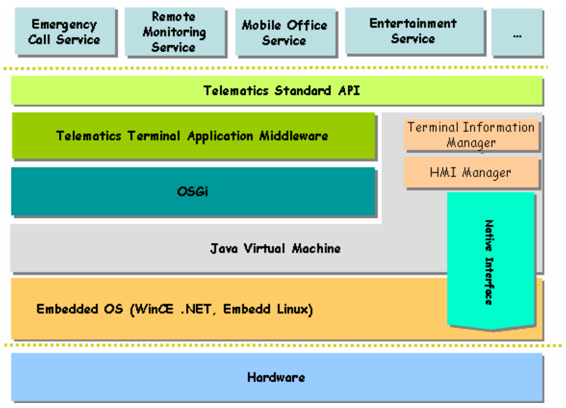

Fig. 1 Telematics Terminal S/W Platform

2.1 OSGi Specification

OSGi is based on the component architecture like CBD (Component Based Development) [1]. There are various types of applications called bundle and are able to use their interfaces only after they are installed and published. And it supports dynamic binding between services over its platform.
OSGi consists of OSGi basis framework and some services such as System Bundle, Package Admin, IO Connector, and so on[2][4]. OSGi basis framework is able to register, search and manage the life-cycle of bundles that are the form of JAR packages. There are six states of bundles in their life-cycle; INSTALLED, RESOLVED, STARTING, STOPING, ACTIVE and UNINSTALLED.

OSGi framework can kill bundles in the run-time anytime but operate and service other bundles that have no dependencies with it continuously.

2.2 AMI-C Specification

AMI-C allows a new digital service to develop telematics services through constructing integrated multimedia device architecture for the automobile information and entertainment system among all types of vehicle in the world. It announced the Release 1 about the overall viewpoint and the conceptual vision to the framework of AMI-C architecture. And then Release 2 introduced a reference implementation to validate the conceptual model, guideline, and specification of network interface in vehicles.
AMI-C API is based on the OSGi framwork and a kind of service implemented the form of OSGi bundle. The API is divided to core services, extension services, and application services. Extension services define address book, telephone book and personal profile. Application services define the off-line navigation. Finally, core services are like as bellows.

- OSGi Service Platform
- Application Execution Management
- Language Management
- Persistent Storage Management
- Vehicle Service Interface
- HMI API

3 Terminal application middleware based AMI-C

The overall structure of embedded S/W platform for telematics terminals we introduce is illustrated in Fig. 2.

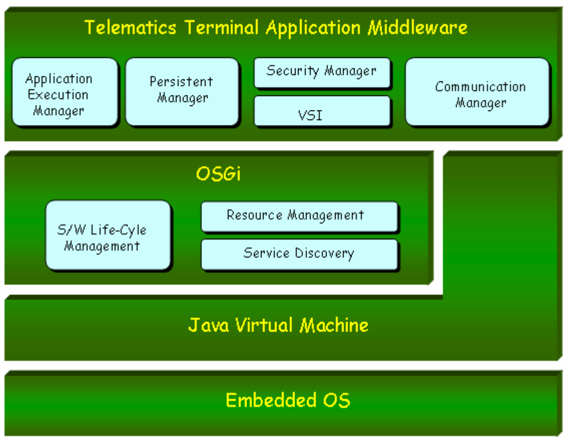

Fig. 2 Telematics Terminal Application Middleware

The core of the S/W platform is a terminal application middleware implemented in accordance with AMI-C specification. Therefore, the middleware takes advantage of OSGi basis framework as an engine to execute applications in distributed environment. The framework includes some functions that are software life-cycle management, resource management, and service management etc. The aim of these functions is to manage JAVA application called OSGi bundle. In addition, the middleware offers vehicles dedicated execution environment which is comprised of various managements such as application execution, persistent storage, communication, security and vehicle service interface.

3.1 Vehicle service interface (VSI)

Terminal information manager collects the states of vehicle, personal information of passenger as well as driver, and terminal data of itself. It has an important role as a gateway between telematics application middleware and internal devices of automobile. According to car makers, there are various models of cars that are used in different protocols and types of devices. Therefore, consistent interfaces or protocols are required to construct a middleware that provides application developers to implement telematics applications.

AMI-C defined uniform interfaces through with a module called vehicle service interface (VSI). VSI can monitor the current status of vehicle and control actuators like open-door lock in automobile. This interface is can either acquire internal states or issue a command through communication manager and ECUs in automobiles. Therefore, it is closely connected to terminal information manager to collect or control vehicle. In case some problems such as the overflow of engine temperature arise, the interface notifies the information to middleware quickly. Fig. 3 shows the relationship between the core modules of telematics terminal application middleware and other sub-systems something like OSGi framework, terminal information manager, ECUs of vehicle, etc.

VSI consists of common message sets that are described a formal language called Abstract Syntax Notation One (ASN.1). Therefore, the messages are network neutral and can applicable to various types of cars with creating a profile like mapping table easily.

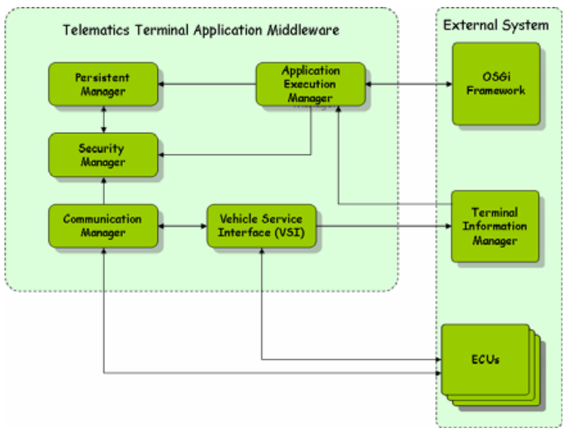
Telematics Terminal Application Middleware
External System
Persistent Manager
Application Execution Manager
OSGi Framework
Security Manager
Communication Manager
Vehicle Service Interface (VSI)
Terminal Information Manager
ECUs

Fig. 3 The relationship with external systems

3.2 Application execution manager

Application execution manager is tightly coupled with OSGi framework and manages telematics application and OSGi bundles installed over terminal S/W platform. As vehicle events are received from terminal information manager, application execution manager can handle special applications which are related to emergency states such as vehicle failure and accident.

Application execution manager execute authentificated applications through security as well as persistent storage manager, and prevent terminal from non-authentificated applications or virus.

Fig. 4 describes a series of process to handle emergency state of vehicle in terminal. Firstly, Terminal information manager managing all information generated an event of vehicle states. And then onUpdate() method in application execution manager automatically is invoked. It reads the state of vehicle through getVehicleState() method in the terminal information manager and control the run level of applications which are operating in the middleware according to the returned state of vehicle. The run level represents a priority related to the execution of application. Applications that is related to monitoring the state of vehicle, controlling vehicle devices as well as outer communication modules like CDMA, and notifying emergency calls can have a high priority. Therefore, if emergency case arises, applications with a low priority will be automatically stopped to get maximum resources in the terminal. And emergency services which have high priorities are invoked and notify the telematics service center(TSP) as soon as possible.

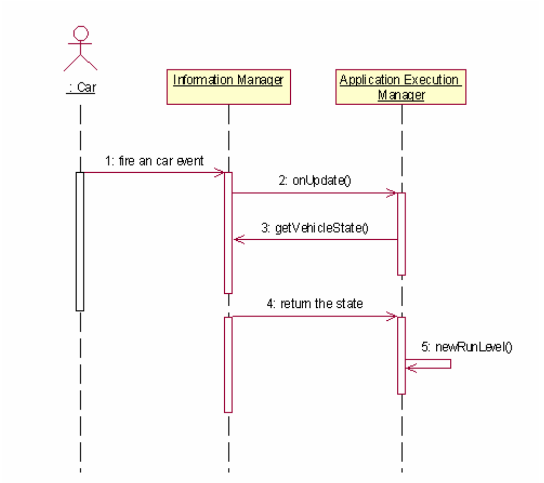

Fig. 4 Sequential diagram for the emergency state

3.3 Persistent storage manager

Persistent storage manager queries and manages the space of storage in the terminal. It also saves data through application execution manager and security manager and protects from non-authentificated applications.

3.4 Security manager

Security manager provides the basic security functions in S/W middleware such as user authorization, execution right, and so on. It also provides the security of personal information from the function of encoding mechanism through the close interconnection with persistent storage manager and protects from hacking through the network with communication manager.

3.5 Communication manager

Communication manager is managing various connections of networks such as CDMA, Bluetooth, CAN, MOST, Serial Port utilized in terminals of vehicle. And Communication manager is compatible and extensible IO Connector interface in OSGi framework. Its interface can be modified and used easily to other types of communication by changing the parameters of URI (Unified Resource Identifier). At presents, the result of this project is supported to CDMA, Bluetooth and Serial communication. And CAN as well as MOST will be developed.

4 Conclusion

Recently, with the help of the technique development of the mobile communication such as CDMA and embedded system, the practical usage of telematics terminal in the mobile environment have been enlarged gradually. At the same time, the necessity and demand for telematics service as well as contents also have been desired so much. But the reusability of the contents comes to fall off because embedded terminal system is optimized in terms of the limited computing power and resources. And the duplication of developments is inevitable and the interoperability among contents and services can not be guaranteed.
Currently, the needs of the standard interfaces of telematics S/W platform increase in developing software for the telematics terminal. Therefore, we implemented a terminal application middleware which played the essential role for the S/W platform.
The middleware of this paper adopted OSGi framework to manage consistently for the all devices connected with the vehicle. In addition, it supports the management of application in the special environment such as vehicles and provides standard programming interfaces to develop telematics contents and services easily.

References:

[1] Open Services Gateway Initiative, "OSGi Service Platform Specification", Release 3, http://www.osgi.org, March 2003

[2] Automotive Multimedia Interface Collaboration, "AMI-C Software API Specification-Core APIs", Release 2, http://ami-c.org, 28 Apr. 2003

[3] Hall R. S., Cervantes H., "An OSGi Implementation and Experience Report", Consumer Communication and Networking Conference(CCNS 2004), 5-8 Jan. 2004, pp.394~399

[4] Joong H. K.; Sung S. Y."Context-aware application framework based on Open Service Gateway", Proceedings. ICII 2001, Vol. 5, 29 Oct. 2001,pp.200 - 204