DESIGN AND IMPLEMENTATION OF A HAVI STACK WITH LINUX AND JAVA

Kuo-Wei Hsu, Chuen-Liang Chen, Ting-Ying Yu, Wu-Cheng Li

Department of Computer Science and Information Engineering
National Taiwan University
Taipei, 106, TAIWAN

ABSTRACT

In this paper, we provide a proposal using Linux and Java to construct the HAVi stack for information appliances. Two big advantages can be obtained from Java use. One advantage is that the transplant nature of programs between platforms is very high, and the other is that the native and Java code can be programmed in parallel. A big advantage of adopting Linux is that developers can use the same operating system both for development and target platforms. The proposed HAVi stack aims at the controller devices that provide the Java runtime environment. We divide the HAVi stack into several layers where each layer contains several functional modules. These se well-designed modules decrease the effort of building this architecture on other platforms or communication media, and they also increase the possibility of introducing new features to HAVi stack.

1. INTRODUCTION

The information appliance (IA) carries many functions, such as a network protocol, the architecture for distributed control, and a graphical user interface (GUI). Therefore, the software realizing these functions will become large-scale and complicated compared with the conventional software for traditional appliances, and it will be expected that a longer period is needed for development.

It is extremely important to develop portable middleware with low cost, because home appliances may be constructed with a lot of heterogeneous platforms. Home network consists of various consumer electronics (CE) devices, and there many different types of processors and platforms are connected together. Java has the advantage of using other languages for implementing the middleware for home network because it has the feature of platform independency and code mobility over network. Moreover, Linux can be easily ported to other platforms besides PC since the Linux community has strong software development ability.

In this paper, we introduce the technique employed to build the middleware, HAVi stack, for IA. We adopt Linux and Java. By the use of Linux and Java, the portability of this kind of middleware can be raised, and it becomes possible to develop application efficiently [1, 2]. Because both Java and Linux run on a variety of platforms, the proposed HAVi stack can be ported to the target platform with very little modification.

The remaining sections are organized as follows. Section 2 introduces the related technologies and standards. Section 3 describes how we benefit from Linux and Java, and further the features of our proposal. Section 4 shows the system architecture for the proposed HAVi stack and finally a conclusion is given in Section 5.

2. TECHNOLOGY BACKGROUND

2-1. HAVi

HAVi (Home Audio/Video Interoperability) is a standard for the interoperability between home appliances, especially for AV appliances. In HAVi, IEEE 1394 is specified as the physical communication media. The HAVi specification [3] defines four kinds of appliances, FAV (Full AV device), IAV (Intermediate AV device), BAV (Base AV device), and LAV (Legacy AV device). FAV and IAV are controller devices used to administer BAV and LAV devices. FAV is equipped with the Java runtime environment and therefore FAV can control other appliances through a Java application.

Following are brief descriptions for each system component of HAVi. The 1394 Communication Media Manager(CMM) helps administer IEEE 1394 network, and also transmits and receives data packets. The Messaging System mediates messages and achieves the communication between software elements. The Event Manager delivers events that present the state change of the network or software elements. The Registry provides the naming service. The Stream Manager provides the connection management of isochronous transmission on the IEEE 1394 bus. The Resource Manager processes the

use demand from each software element. The Device Control Module (DCM) is used to control an appliance connected to the network. HAVi ensures interoperability by standardizing the control of device functionality in the Functional Component Module (FCM). It abstracts each function in appliances. Examples of FCMs are camera, tuner, and amplifier FCM. HAVi application can control devices by accessing their FCMs. The DCM Manager performs installation and uninstallation of DCM and FCM.

In the HAVi network, the controller software residing on each HAVi device provides services to the applications. A HAVi application acts as a HAVi client. Services are accessed through typical client-server interactions. The client application operates with the invocations of standard HAVi API. The information of such invocation is embedded within a HAVi message and sent to the specific HAVi system component or a proper software element through the Messaging System. The server extracts the original call, carries it out, and returns the results back to the client through the Messaging System.

2-2. IEC 61883

IEC 61883 defines a digital interface for IEEE 1394-enabled AV devices. It is composed of five parts. Only the first one is adopted to construct HAVi stack. It defines the function control protocol, connection management procedures, and also the format of common isochronous packet and plug control register.

2-3. IEEE 1394

IEEE 1394 is an industry standard defining a high speed serial bus that runs at speeds of up to 400. Once some device becomes IEEE 1394-enabled, it can share streaming data with other devices or be controlled by them. IEEE 1394 devices can be accessd under Linux [4]. Figure 1 illustrates the Linux 1394 subsystem and the middleware that run on it.

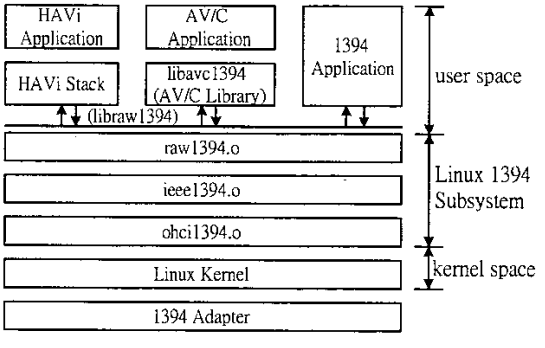

Figure 1: Linux 1394 subsystem

III. SYSTEM FEATURES

3-1. Benefits from Java

The most significant feature of Java language is "write once and run anywhere". The Java virtual machine (JVM), which is the basis of the runtime environment, is implemented on a variety of platforms, and therefore Java programs can run on various platforms. From the high reusability of code, Java is used for software extensions in HAVi. Each HAVi device offers a control module that includes a user interface and a specific DCM. This control module could be a Java bytecode file (code unit in JAR format) that is uploaded to and executed on the controller devices.

Two big advantages can be obtained from using Java. One advantage is that the transplant nature of programs between platforms is very high, and the other is that the native and Java code can be developed in parallel.

3-2. Benefits from Linux

The main feature of Linux is that it is an open source platform. Therefore, as a result of improvement being performed by many developers, Linux is with very high stability. The rich programming environment provided by the Linux makes the software development easy. Linux is already ported to many platforms. Linux adding improvement for specific uses can also come to hand. The biggest advantage of using Linux is that developers can use the same operating system both for development and target platforms. Therefore, a program built on a development machine can be executed on a target appliance without modifying it.

IV. SYSTEM ARCHITECTURE

4-1. Overview

System architecture for the proposed HAVi stack is shown in Figure 2. In order to increase the portability, we divide the system into several layers where each layer depends only on its underlying layer and can be replaced alone. The bottom layer is the hardware platform. Up the hardware platform is the Linux operating system and its 1394 subsystem. Linux 1394 subsystem acts like an IEEE 1394 stack that consists of the driver for IEEE 1394 adapter and the programming library. The 1394 stack API offers interoperability to upper layers because of the high portability of Linux.

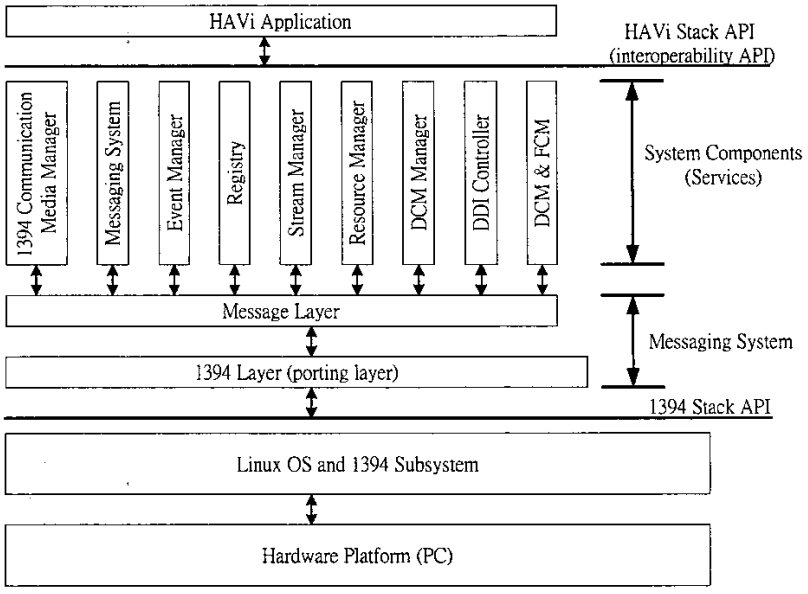

Figure 2: System architecture

It is up to the 1394 Layer, which uses the 1394 stack API, to offer Java programs the access capability for IEEE 1394 network. Up the 1394 Layer is the Message Layer that implements the high-level transmission functions of the Messaging System. These transmission functions are independent of IEEE 1394 while those dependent functions are implemented in the 1394 Layer. The Message Layer also implements the HAVi RMI (Remote Method Invocation) protocol, which is a high-level communication mechanism between software elements and system components. System components, which provide system services defined in specification, are based on the Message Layer. HAVi stack ensures the interoperability through a set of standard APIs. APIs of HAVi stack are defined in IDL (Interface Definition Language in [5]) but the implementation is free for developers. Instead of specifying the native binding, HAVi defines a Java binding called HJA (HAVi Java API) for uploaded DCMs [6]. On the top is the HAVi application. One HAVi application is called a havlet if it is implemented with Java.

4-2. 1394 Layer

The native portion depending on platform has to de developed in languages other than Java, such as C language. All elements of the proposed HAVi stack are implemented as Java classes except those in the 1394 Layer. The 1394 Layer, which is based on Linux 1394 subsystem, contains several Java classes that are designed as a set of wrapper classes for the native functions. It can be viewed as an abstraction layer. Classes in the 1394 Layer package up the functions provided by Linux 1394 subsystem. They contain native methods that invoke the native functions through the Java native interface (JNI).

In HAVi specification, two types of services are provided by CMM. One service is to provide a transport mechanism to send to and receive packets from remote devices. The other is to present the network activities and information to other software elements. It is obvious that the former is media dependent but the latter is not. We abstract the fundamental transmission functions from CMM and place them in the 1394 Layer. In our proposed architecture, CMM only provides high-level functions that are media independent. This design increases the portability, and it also provides the possibility for HAVi stack to be established over another communication media.

4-3. Messaging System

The Messaging System provides all software elements with communication facilities. It is composed of the message layer and the Transport Adaptation Module

(TAM). In our proposal, TAM uses the 1394 Layer to send and receive messages.

In the proposed architecture, the job of every system component is to provide services to applications. HAVi applications and system components are communicated with each other through HAVi RMI. The communication progressed inside the stack is not defined in HAVi specification, and in our proposed architecture it is achieved through procedure calls. In our Messaging System, TAM is composed of several Java classes. And also, we abstract the messaging section from the Messaging System. There four threads run in the Messaging System. One thread in the Message Layer is used to transmit messages, one in the 1394 Layer to receive 1394 packets, one in the Message Layer to receive messages, and one in the Message Layer to deliver messages.

4-4. System components

Services provided by system components are defined in specification and implemented as sets of functions. HAVi RMI is a client-server (request-response) communication mechanism. A system component can be viewed as a server that processes the request messages sent by clients. An application or DCM can be viewed as a client that sends request messages to the server and then processes the responses returned from the server. HAVi RMI is based on the standard message format, and in both the request and response message there is a field reserved for indicating how this message should be processed. In order to ensure the interoperability, HAVi defines a set of operation codes for each service. Figure 3 represents the interoperability API of HAVi. In HAVi, functions used to process requests can be implemented with any programming language. We provide Java binding for these functions since we focus on the uploaded applications and DCMs and the proposed HAVi stack aims at the FAV fields.

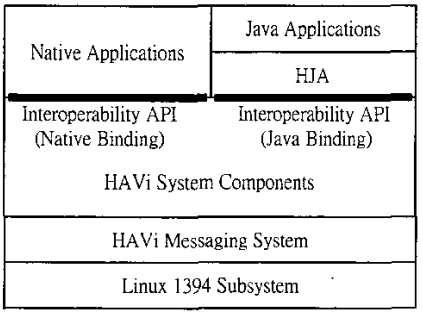

Figure 3: Interoperability API

System components, applications, and DCMs are software elements. Class *SoftwareElement* is used by an application or DCM, while *SystemComponent* is an abstract class that helps the implementation of system components and also helps the development of new services outside specification. In order to reduce the overhead caused by creating threads, these two classes are designed the as data classes rather than Java threads.

V. CONCLUSION

In this paper, we describe how to use Linux and Java to develop the HAVi stack for information appliances. The proposed architecture is based on Linux and Java. One benefit form Java is that the transplant nature of programs between platforms is very high, and the other is that the native and Java code can be programmed in parallel. On the other hand, a big advantage of adopting Linux is that developers can use the same operating system both for development and target platforms. From the achievement of the proposed HAVi stack, we can say that using Linux and Java for development has worth of examination much.

REFERENCES

[1] Kouta Soejima, Masahiko Matsuda, Toru Iino, Taketoshi Hayashi and Tatsuo Nakajima, "Building Audio and Visual Home Appliances on Linux", in Proc. *Proceedings of the 2002 Symposium on Applications and the Internet (SAINT)*, 2002.

[2] Tai-Yeon Ku, DongHwan Park and Keong -Deok Moon, "A Java-Based Architecture Supporting IEEE 1394 for Home Entertainment Network", *International Conference on Consumer Electronics (ICCE)*, 2002.

[3] The HAVi Organization. (2000, January 18). *The HAVi Specification.* (version 1.0), http://www.havi.org/techinfo/index.html

[4] The Linux 1394, http://www.linux1394.org/

[5] The Object Management Group (2001, February 1). *The Common Object Request Broker: Architecture and Specification, Chapter 3: OMG IDL Syntax and Semantics.* (version 2.4.2), http://www.omg.org/technology/documents/formal/corbaiiop.htm

[6] Rodger Lea, Simon Gibbs, Ravi Gauba, Ram Balaraman, and Eddy Odijk, *HAVi Example By Example: Java Programming for Home Entertainment Devices*, 1st edition, Prentice Hall, August 2001, chapter 3.