A Study on Mobile Virtualization

Hyun-suk Roh*, Hyun-woo Lee*, Sang-ho Lee**

*Electronics and Telecommunication Research Institute(ETRI), Daejeon, Korea

**Computer Science Department, Chungbuk National University, Cheongju, Korea

ventus@etri.re.kr, hwlee@etri.re.kr, shlee@cbucc.chungbuk.ac.kr

Abstract— **This paper proposes a mobile virtualization system based android-x86 virtual machine. The proposed system provides virtual services to a light-weighted device. Those are provided from the android-x86 based virtual machine. Here, light-weighted device is like a phone or a pad just having the ability of H.264 decoding and a virtual machine is based android-x86 using Oracle VirtualBox. In this paper, we propose several methods that android-x86 based applications can operate same way as it does in real smart devices. Those include device virtualization and hardware acceleration schemes.**

Keywords— **virtualization, virtual machine, android, x86, openGL**

I. Introduction

Cloud computing known as first proposed by Google was focused by the world as IT, economic journals and representative Global enterprise CEOs emphasized it. That way, cloud was topic of market. But with the mobile shortly after it is combined, 'Mobile Cloud Computing' is new topic of the market which leads main streams of the cloud computing.

If users can connect to the network, Cloud computing make users can share data and collaborate with each other, as well as computing resources can be utilized efficiently regardless to time and place. So, cloud computing is variously used from storing private data at web hard to collaboration between multinational corporations. As mentioned above, Mobile cloud computing, as it is necessary to use and it used as it pays, is a combination of cloud computing and mobile. Mobile is various devices with mobility function including smart phones.

The mobile cloud computing is spreading to various fields, but platform and standardization related issues are constantly emerging. Mobile platform is largely divided into terminals platform and server platform. Terminal platform is such things mounted on devices like operating system, middleware and browser. Server platform is such things mounted on the server like authentication, billing, gateway and online market place.

The first platform is the ability to control the hardware, stick to the UI (user interface) that were insufficient to directly control the hardware platform. So it is difficult to aware the platform but recently, with the improvement of the performance of the terminal and the development of the UI, users has evolved to a level which can access easily to the platform. As a result, users equipped high performance terminals in using them and had been asked to manipulate more high level UI.

In addition to the use of these high-end terminals, recently light-weighted platform technology has been proposed which provides high quality web/app service to various portable terminal platforms without any hardware performance.

These technologies include a contents virtualization without any dependencies with smart web and app contents, cloud light-weighting service to support various terminals and service adaptation on contents consuming environment.

The remainder of this paper is organized as follows. We briefly introduce the mobile virtualization system in Section 2. Section 3 describes the proposed technologies for mobile virtualization system. This section includes virtual device driver and OpenGL based hardware acceleration and finally, we draw our conclusions in Section 4.

II. Mobile Device Virtualization System

A Mobile virtualization system is divided mobile devices and virtualization server described in Figure 1.

Mobile devices and virtualization server are connected by Wi-Fi basically and additionally 3G or LTE are possible. Mobile devices, low performance and light-weighted devices, have minimum codecs and sensors and only have the decoding function which can decode the data from the server.

The virtualization server based cloud system have android-x86 virtual machines which receive events from mobile devices and send encoded screens to mobile devices.

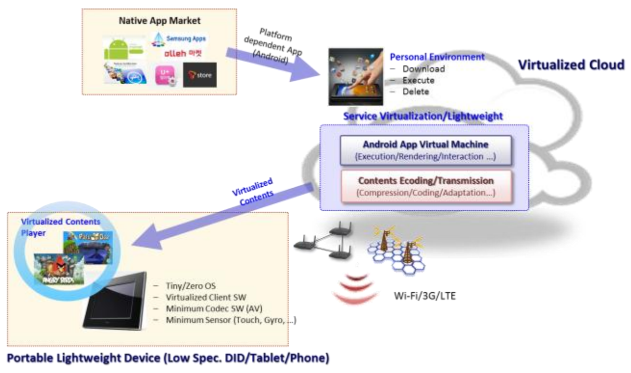
Native App Market
Platform dependent App (Android)
Personal Environment
– Download
– Execute
– Delete
Virtualized Cloud
Service Virtualization/Lightweight
Android App Virtual Machine
(Execution/Rendering/Interaction ...)
Contents Ecoding/Transmission
(Compression/Coding/Adaptation...)
Virtualized Contents
Virtualized Contents Player
– Tiny/Zero OS
– Virtualized Client SW
– Minimum Codec SW (AV)
– Minimum Sensor (Touch, Gyro, ...)
Wi-Fi/3G/LTE
Portable Lightweight Device (Low Spec. DID/Tablet/Phone)

Figure 1. mobile virtualization service

III. Virtual Server System

A. *Android Software Structure*

Figure 2 show the software structure of android. Linux kernel is located at the bottom of the android system and hardware related library, C/C++ libraries, application framework and applications exists above the linux kernel in order.

Because there is no physical GPS and no sensors, android-x86 cannot recognize transmitted values from the device. In order to solve this problem, we need virtual paths to android framework. This is virtual device drivers and HAL layer. We made virtual driver and HAL layer about each event from GPS and sensors.

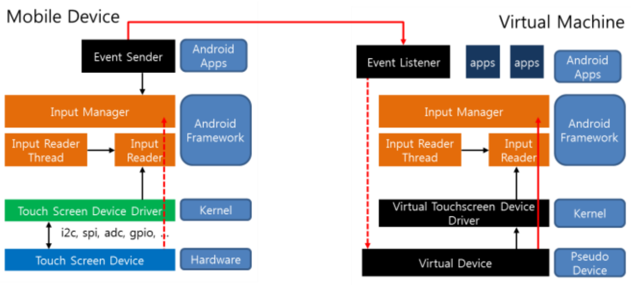

Figure 2. device events management

B. *Virtual device driver*

Driver virtualization technology in A Mobile virtualization system is divided virtual device drivers and HAL(Hardware Acceptance Layer) libraries.

Virtual device drivers are divided into virtual sensor drivers and GPS driver again. Virtual sensor driver transfer transmitted sensor values to android frame work and virtual GPS driver transmitted GPS NMEA data to android frame work upward direction through HAL layer.

HAL libraries connecting the hardware and android framework are required to access the hardware of linux kernel based. As described prior, our mobile virtualization system requires interface library layer to transfer event values of devices(GPS, sensor data) to android application layer. That is HAL libraries.

C. *OpenGL ES based Hardware Acceleration*

Because arm core based application need hardware acceleration applications using arm core cannot be executed in mobile virtualization system, x86 based virtual machine. So we should provide another environments. Our methods are using OpenGL acceleration. When system need hardware acceleration, our application activates openGL interfaces which are connected host's openGL libraries outside of the virtual machine. Next, host OS displays hardware acceleration results on screen.

Using this mechanism, we improve the performance of specific games and browsers.

Figure 3. virtualization execution window

1) ***Overall Structure***: Our overall structure is described in Figure 5. Screen information transmitted from applications are managed by surface composer which calls openGL interface. openGL interface and Host OS are connected each other through VirtualBox Tunnel and Host openGL hardware is used by Host openGL Library.

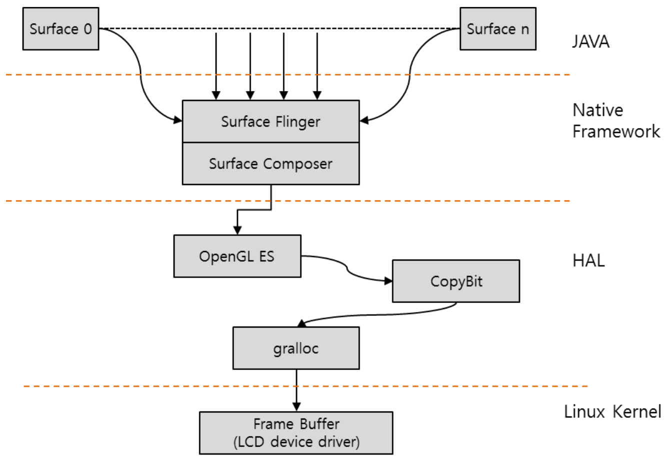

Figure 4. general display structure

Figure 4 describes general components and operating procedure. General system uses openGL ES mechanism which call real hardware related libraries. This means that general system has real display related device and use it when decoding display data. But our system has no real display related device. So we need some other method.

In our system, surface composer is a combination of upper layer's applications and OpenGL Interface is HAL libraries made by C/C++ codes. Between Virtual Machine and Host OS, there is a path which is a bi-directional way from virtual machine and host OS.

In our system, we also provide an additional encoding server which can exist in outside system. This method improve our system performance and allow more users to get services.

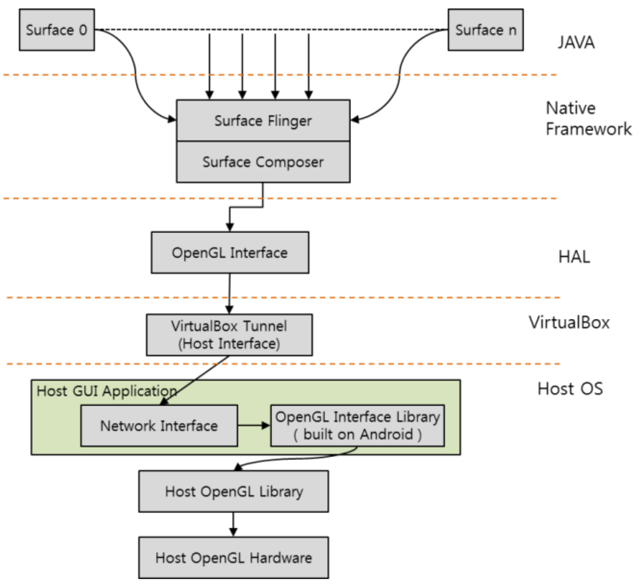

Figure 5. device events virtualization structure

2) ***Execution Procedure***: First, using Run script, we execute a headless virtualbox and load GUI application in the Host OS. Headless virtualbox boots android and calls a openGL interface and a GUI application creates window and connect with a openGL interface. Then using openGL interface library, host OS displays openGL results to the screen.

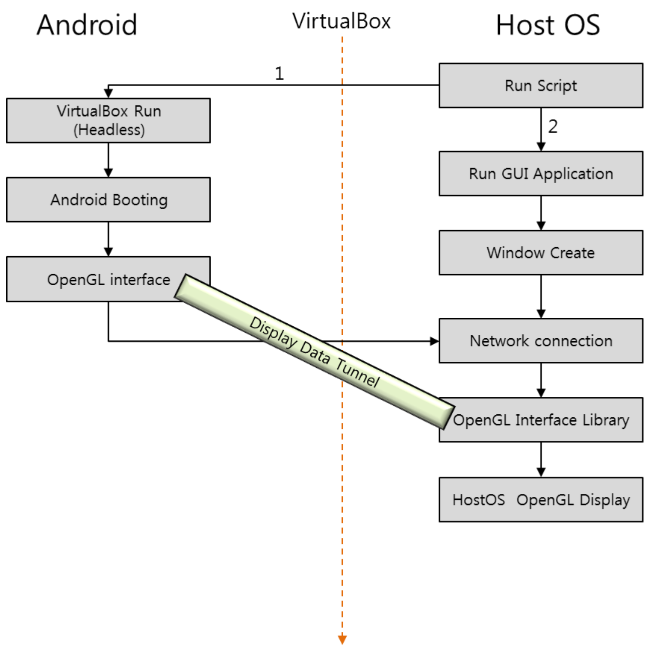

Figure 6. Device events virtualization procedure

IV. Conclusions

In this paper, we proposed a mobile virtualization system. We need new node like Location Server and Paging Controller. We explained problems for mobile virtualization of android-x86 and suggest solutions. One is device virtualization and the other is OpenGL ES based hardware acceleration.

Using proposed schemes we can provide various smart services to light-weighted devices. In the future, we will expand our schemes to multiple user virtualization environments.

Acknowledgment

This research was supported by the MSIP (Ministry of Science, ICT & Future Planning), Korea, under the R&D program supervised by the KCA (Korea Communications Agency) (KCA-2013-12-912-03-001)

Corresponding author of this paper is Sangho Lee (shlee@cbucc.chungbuk.ac.kr) at Chungbuk University.

References

[1] A. Charland and B. LeRoux, "Mobile application development: Web vs.
[2] native," Queue, Apr. 2011, pp. 20-28.
[3] Ki-Hoon Lee, Dong-Hoon Kim, and Gyu-Tae Baek, "Web application
[4] virtualization for IPTV," Proceedings of Network Operations and Management Symposium, Sept. 2012, pp.1-4.

[5] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster, "Virtual infrastructure management in private and hybrid clouds," IEEE Internet Computing, 2009, vol. 13, no. 5, pp. 14-22.
[6] Web site : http://developer.android.com
[7] Web site : www.androvm.org

Hyunsuk Roh received the B.S. and the M.S. degrees in Computer Engineering from Kyungpook National University, Dae-gu, Korea, in 1995 and 1997, respectively. He was a R&D Engineer at the Network Research Lab. Since 2000, he has been a Staff R&D engineer at the Electronics and Telecommunications Research Institute (ETRI), Korea. His current research interests include Mobile Virtualization, Cloud Virtualization and Multicast/Broadcast Technologies.

Hyunwoo Lee received the B.S., M.S. and the Ph.D. degree in the School of Electronics, Telecommunication and Computer Engineering at Korea Aerospace University, Korea in 1993, 1995 and 2005 respectively. From 1995, he has joined Electronics and Telecommunications Research Institute (ETRI). From 2009, he is a team leader in the Smart Screen Convergence Research Department in ETRI. His research interests include IPTV service, service control technology, and traffic congestion control.

Sang-Ho Lee was born in PuSan, Korea in 1953. He received the B.S. degree in the department of computer science, Soongsil University in February 1976. He received the M.S. degree and Ph.D in the department of computer science, Soongsil University in February 1981 and 1989 respectively. He is currently working professor in the department of software engineering, Chungbuk National University. His research interests also include Protocol Engineering, Network Security, Network Management and Network Architecture.