Android Phone as Wireless USB Storage Device through USB/IP Connection

Kyuchang Kang, Dongoh Kang, Kiryong Ha and Jeunwoo Lee, *Member, IEEE*

***Abstract*-- This paper addresses an Android phone as a wireless USB storage device through USB/IP connection. For this expanded function, we need to make backing storage with phone storage such as SD card and translate USB file system by using file-backed storage gadget. To connect wirelessly with a host computer, USB/IP gadget transmits data through IP layer. The goal of this paper is to define an additional function of a smart phone operated as a wireless USB storage device and suggests the usage scenario in virtual computing environment estimated as future home computing infrastructure based on virtualization technology.**

I. INTRODUCTION

Today, smart phones are not exclusive property of early adopters or IT professionals.

Global smart phone shipments grew a relatively healthy 43 percent year-over-year to reach 60 million units in Q2 2010 [1].

According to comScore's report [2], 234 million Americans subscribed to mobile phone plans in January 2010. Of these, 42.7 million owned Internet-accessible smart phones, which represented an 18 percent increase over the three months ended in October.

Barton et al.[3] propose that important progress in mobile computing should start with straightforward local convergence of smart phones and computer infrastructure via simplistic mobile use model. This use model for the computing aspect of mobile phones would build off the use model for auxiliary storage, specifically the USB memory stick.

In Android market, we can find several applications enabling the storage of phone to be used as an USB storage device. MountUSB [4] is a one-click tool which allows to mount or unmount SD card as an USB mass storage device. USB Switcher [5] can mount a SD card in Android phone on a host computer.

These applications provide a SD card with USB connectivity but we always need a connecting cable. In addition to the function of storage connectivity, we need to allow for wireless connection for a convenient user experience because we may not always carry a connection cable with a phone.

Therefore, we propose wireless connecting function of SD card of a phone which enables Android phone to be wireless USB storage device.

A rest of the paper is organized as follows. In section II, we describe underlying technique and implementation for wireless USB connection in detail. The section III presents enabling usage scenarios and remarks. Finally, we summarize and address future works in section VI.

II. IMPLEMENTATION

For enabling Android phone as a wireless USB storage device, we need two major steps. First of all, we should select some parts or all of SD storages in the phone and emulate them with USB device. Next, we should plug wirelessly the emulated USB device into a host computer as if general USB memory sticks are plugged into the host computer.

A. Overall architecture

Fig. 1 shows architectural diagram and inner modules for wireless USB storage.

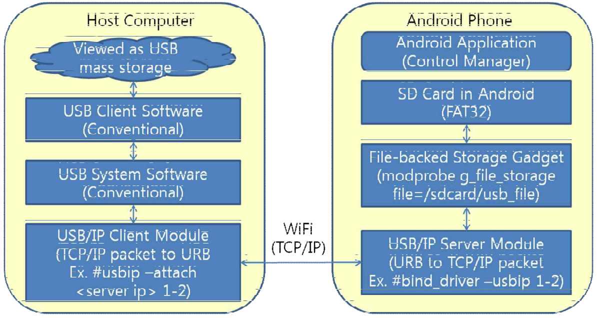

Host Computer
Viewed as USB mass storage
USB Client Software (Conventional)
USB System Software (Conventional)
USB/IP Client Module (TCP/IP packet to URB Ex. #usbip –attach <server ip> 1-2)
WiFi (TCP/IP)
Android Phone
Android Application (Control Manager)
SD Card in Android (FAT32)
File-backed Storage Gadget (modprobe g_file_storage file=/sdcard/usb_file)
USB/IP Server Module (URB to TCP/IP packet Ex. #bind_driver –usbip 1-2)

Fig. 1. Architectural diagram for wireless USB storage.

B. Underlying technique

In order to translate files of SD card, we use file-backed storage gadget [6]. For file translation, we need to create a backing storage file. Backing storage in advance to translation requires some preparation before the gadget can use it. If the backing storage is a regular file in translating process, then the file must be created beforehand with its full desired size. Because the gadget won't create a backing storage file and won't change the size of an existing file automatically.

As a USB connection over network, we use open source project, 'USB/IP Project' [7]. This project aims to develop a general USB device sharing system over IP network. To share USB devices between computers with their full functionality, USB/IP encapsulates USB I/O requests into IP packets and transmits them between computers.

C. Android programming

The software module for a phone shown in Fig. 2 consists of application program (.apk) and native module (.exe) because of permission policy of Android application that an application cannot have root privilege for executing kernel level operation.

The application module interacts with a user and dispatch

This work is supported by the IT R&D program of MKE/KEIT [2008-S-034-01, Development of Collaborative Virtual Machine Technology for SoD (System on-Demand) Service]

control messages to the native module. After receiving control messages, the native module interacts with kernel to execute kernel mode operation such as ‘insmod’.

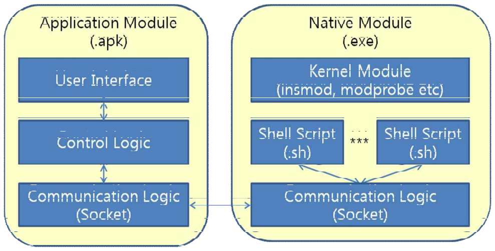


Fig. 2. Software modules for control manager in the phone.

D. Results

The emulated USB storage of Android G1 phone has a plain text file, wave file, and image file. These files are accessed and played in a host computing device through USB/IP connection as if those files saved in the host computing device.

Fig. 3 shows implemented components and used underlying modules. There are an Android application as a GUI module and two shell files in order to start or stop USB/IP operation.

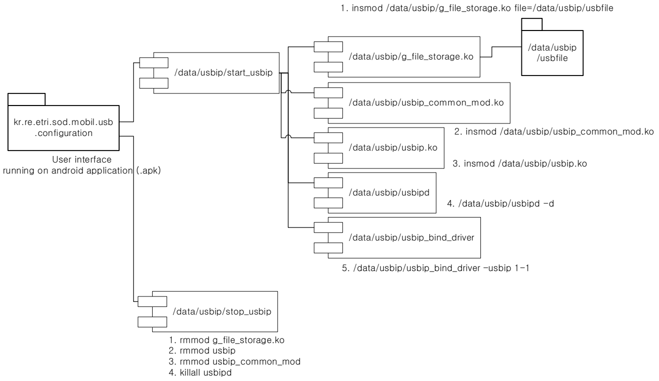
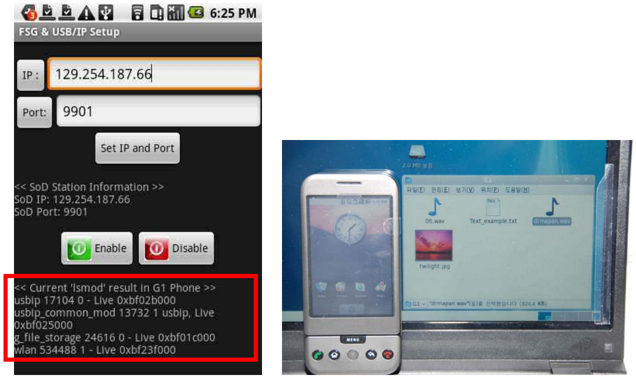

1. insmod /data/usbip/g_file_storage.ko file=/data/usbip/usbfile
/data/usbip/g_file_storage.ko
/data/usbip
/usbfile
/data/usbip/start_usbip
kr.re.etri.sod.mobil.usb
.configuration
User interface
running on android application (.apk)
/data/usbip/usbip_common_mod.ko
2. insmod /data/usbip/usbip_common_mod.ko
/data/usbip/usbip.ko
3. insmod /data/usbip/usbip.ko
/data/usbip/usbipd
4. /data/usbip/usbipd –d
/data/usbip/usbip_bind_driver
5. /data/usbip/usbip_bind_driver –usbip 1–1
/data/usbip/stop_usbip
1. rmmod g_file_storage.ko
2. rmmod usbip
3. rmmod usbip_common_mod
4. killall usbipd

Fig. 3. Component diagram for operation.



IP : 129.254.187.66

Port: 9901

<< SoD Station Information >>
SoD IP: 129.254.187.66
SoD Port: 9901

```
<< Current 'lsmod' result in G1 Phone >>
usbip 17104 0 - Live 0xbf02b000
usbip_common_mod 13732 1 usbip, Live
0xbf025000
g_file_storage 24616 0 - Live 0xbf01c000
wlan 534488 1 - Live 0xbf23f000
```

Fig. 4. Results of prototype.

Fig. 4 shows the screen shot of Android application displaying the result of ‘lsmod’ Linux command and the picture of Linux host computer describing connected files of plugged USB device wirelessly.

III. USAGE SCENARIOS AND REMARKS

Implemented prototype can be used to wireless USB storage for Linux and Window machine. It can be also applicable to wireless USB storage for guest virtual machine generated and allocated dynamically in virtualization computing environment. In other words, this can be used to device cooperation to build computing environment by exposing their storage functions as a form of virtualized USB device, especially USB storage device, to operating system.

As a limitation of prototype, because of the bandwidth limitation of 54Mbps Wi-Fi environment, a large volume of storage is not adequate currently through USB/IP protocol.

IV. CONCLUSION

This paper addresses the current distribution of smart phone and the needs for additional function from usage model perspective. As an added function of conventional smart phones, we propose the method how to transfer an Android phone into a wireless USB storage device through USB/IP connection and show the prototype implementation.

The goal of this paper is to define an additional function of a smart phone operated as a wireless USB storage device and suggests the usage scenario in virtual computing environment estimated as future home computing infrastructure based on virtualization technology.

We need to allow for following issues as the future considering works to evolve into a full-fledged system. Firstly, we may consider various operating systems used for terminals because the prototype is implemented on Linux including Android. Secondly, we also allow for performance issues while a storage connection through IP network.

REFERENCE

[1] M.J. Duftler, N.K. Mukhi, A. Slominski, and S. Weerawarana, “Web services invocation framework (WSIF),” OOPSLA 2001 Workshop on Object-Oriented Web Services, 2001.

[2] F. Villanueva, D. Villa, M. Santofimia, F. Moya, and J. Lopez, “A framework for advanced home service design and management,” Consumer Electronics, 2009. ICCE '09.

[3] A. Gárate, N. Herrasti, and A. López, “GENIO: an ambient intelligence application in home automation and entertainment environment,” Proceedings of the 2005 joint conference on Smart objects and ambient intelligence, ACM, 2005, pp. 241-245.

[4] VIA Artigo Pico-ITX Builder Kit, http://www.via.com.tw/en/products/embedded/artigo/a1000/index.jsp

[5] R. Van Engelen, K. Gallivan, G. Gupta, and G. Cybenko, “XML-RPC Agents for Distributed Scientific Computing,” IN IMACS“2000 CONFERENCE, 2000, pp. 168-182.