Android-based SoD Client for Remote Presentation

Kyuchang Kang*, Kiryong Ha*, Jeunwoo Lee*
*Software Research Lab., Electronics and Telecommunications Research Institute
161 Gajeong-dong, Yuseong-gu Daejeon, Korea
{k2kang, lyonha, ljwoo}@etri.re.kr

Abstract—This paper presents Android-based SoD (System on-Demand) client for remote presentation in virtual desktop environment. SoD is framework for on-demand computing in virtual computing environment. SoD enables to build cooperative device collaboration by orchestrating virtualized peripheral resources such as monitors, keyboards, mice and so on. The objective of this paper is providing an Android-based phone with SoD client function, especially, remote presentation including functions of a mouse, a keyboard and a monitor. Implemented SoD client is tested on virtual desktop environment consisting of a Xen hypervisor, Windows-based guest virtual machines. The challenge point of this paper is separating and virtualizing traditional peripherals of desktop. After this, we can redirect each virtualized I/O to any SoD client device on demand.

Keywords—System on-demand, remote presentation, remote I/O, I/O virtualization, virtual desktop

I. INTRODUCTION

Currently, the future personal computing environment is expected to be led by virtualization-based cloud computing and smartphone-based mobile computing. This development of cloud computing technology is realizing SoD (system on-demand) service which configures virtual computing environment with ubiquitous devices. Meanwhile, the boom of smartphone accelerates various mobile computing services.

Due to these trends, desktop virtualization [1] has the potential to bring about a fundamental change in end-user computing environment. As the proliferation of desktop virtualization, thin client terminal is a growing need. On the other hand, the rapid growth of smartphone is rapidly replacing the role of thin client terminal.

However, current desktop virtualization solutions [2, 3] have a limit to provide configurable I/O devices with virtual desktops dynamically.

From the point of using a virtual desktop, we need to select or unselect his/her devices such as a smartphone as a peripheral of the virtual desktop. Therefore, we address in this paper a SoD client providing a function of dynamically configurable peripheral devices for a virtual desktop, especially, focusing on SoD client based on Android platform as a client part of SoD service environment.

This technique is one of solutions enabling computing resources such as a virtual machine to connect with distributed peripheral devices such as a mouse, a keyboard, a speaker, or a microphone.

The SoD client scheme can contribute to realizing the VPI (virtual peripheral interface) enabling resource fragmentation and composition on demand.

The rest of the paper is organized as follows. In section II, we explain the background environment of this paper such as SoD service and related works on remote presentation. The section III presents the SoD client architecture and the function of software components. The section IV describes prototype design and its operation in detail. The section V presents the operation environment and the result of prototype implementation. Finally, we summarize this paper and future considerations are mentioned in section VI.

II. BACKGROUND

This section presents the background environment for the SoD client and related works on remote presentation which is the main function of proposed SoD client.

A. SoD Service and Execution Environment

SoD service provides users with customized computers through network. The customized computers are comprised with user-preferred operating system, applications and peripheral devices through dynamic integration of them.

The users of SoD service can comprise and leverage the computer easily after accessing the
server (SoD server) with a web browser or a dedicated application of the mobile device and comprising them by means of Pick & Drop fashion. Therefore, the users can leverage computers having expected performance and functions flexibly at anytime and anywhere through collaboration of mobile devices and peripheral I/O devices.

The SoD zone is defined as the area that provides computing environment including computing resources, I/O resources including distributed peripheral devices.

A SoD zone consists of a SoD station, SoD storage servers, and SoD servers. The SoD station is orchestrating zone computing environment by allocating/deallocating computing resources such as I/O peripherals, virtual machines, and users through a resource management module as a zone control center. The SoD storage server is an image server such as NAS (network-attached storage). It contains private or public virtual machine and user disks. The SoD server is similar the SoD station but has slave function of control center. This is for scalable computing resources and is expended on demand.

B. Remote Presentation

In Android market, we can find several applications enabling remote presentation [4-8]. Typical solutions are based on RDP (remote desktop protocol) [9] such as Remote RDP [4] and Remote Desktop Client [5] or VNC (virtual network computing) [10] such as Remote VNC Pro [6] and Andorid-vnc-viewer [7].

Citrix Receiver [8] is a proprietary protocol and it delivers computing as a service.

These applications provide a computing user with remote presentation of current his/her computing including a virtual desktop. However, these applications cannot separate resources that consisting remote presentation such as a keyboard, mouse and monitor resources. In addition, theses applications do not support the configuration of each resource freely.

III. SoD CLIENT ARCHITECTURE

This section describes SoD client architecture and its function.

A. Main Functions

The main functions of SoD client are system login, resource allocation/deallocation and resource virtualization shown in Table I.

<table>
<thead>
<tr>
<th>Function</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SoD System Login</td>
<td>Authenticate users through XML-RPC communication</td>
</tr>
<tr>
<td>Resource allocation/deallocation</td>
<td>Receive list for currently existing virtual machines and virtual resources</td>
</tr>
<tr>
<td></td>
<td>Execute the virtual machine through user’s input</td>
</tr>
<tr>
<td></td>
<td>Allocate resources through user’s input</td>
</tr>
<tr>
<td></td>
<td>Deallocate resources through user’s input</td>
</tr>
<tr>
<td>Resource virtualization</td>
<td>Keyboard Transmit keyboard inputs to the server</td>
</tr>
<tr>
<td></td>
<td>Mouse Transmit touch screen inputs to the server</td>
</tr>
<tr>
<td></td>
<td>Monitor Receive frame buffer data of server and draw on the screen</td>
</tr>
</tbody>
</table>

B. Software Architecture

SoD client software consists of XML-RPC module, Command Controller module and each resource modules shown in Figure 1.

Through communication with XML-RPC server of SoD station, the XML-RPC module authenticates users and processes following works:

- Receiving the list of resources and virtual machines
- Allocating/deallocating the resource
- Updating the list of resources
• Starting/stopping virtual machine (VM)

In addition, whenever the status change of the resources, the XML-RPC module listens the event. Apart from resource allocation/deallocation, the Command Controller is responsible for control protocol to virtualize the resource of phone. Therefore, the Command Controller should transmit the phone’s resource information to VDM (virtual driver manager) whenever the client program is starting.

From the resource description points, we consider various peripherals as USB-like devices because most of peripherals support USB connectivity and its description is intuitive. Therefore, each of resource information about peripherals should follow the USB descriptor. However, the phone is not USB device that the information about usb_speed, usb_version, vendor_id and product_id are set as 1, 1.1, 0x0000 and 0x0000 for each.

Things to watch are that bus_numbers for each monitor, mouse and keyboard should be different. The vendor name and product name are represented as string value.

The Command Controller processes and responds the VDM messages about resource allocation/deallocation as followings:

• VP_REQUEST_DEVICE: request message for a specific resource to be allocated to the given VM
• VP_RELEASE_DEVICE: request message for a specific resources to be deallocated to the given VM

The monitor module displays the frame buffer data of VM on touch screen after receiving them from the SoD station. Previously, the monitor module performs TCP connection to the VM after receiving VM’s information from Command Controller.

Because the VM’s resolution is over 1024x768 in general, we allow for configurable resolution of the client to support zoom-in and zoom-out through multi-touch in system design.

The mouse module performs TCP connection to the VM and loads mouse pad on the screen of user’s phone after receiving VM’s information from Command Controller. Then, it transmits the movement information entered by the user to the connected VM.

Add to basic mouse input, we allow for touch input as followings:

• One finger tap: left button click of a mouse
• Two fingers tap: right button click of a mouse
• Double taps: double clicks of left mouse button

The keyboard module performs TCP connection to the VM and loads keypad on the screen of user’s phone after receiving VM’s information from Command Controller. Then, it transmits the key code information entered by the user to the connected VM.

IV. DESIGN AND OPERATION

This section describes operating procedures and defined message protocols for prototype implementation.

Firstly, Figure 2 shows the operating procedures for connection and termination of resource-providing devices by executing or terminating SoD client application.

Figure 2. Operating procedures for connecting and termination devices
Figure 3 shows the operating procedures between Android phone and SoD station for resource allocation.

![Figure 3. Operating procedures for resource allocation](image)

Figure 4 shows the operating procedures between Android phone and SoD station for resource deallocation.

![Figure 4. Operating procedures for resource deallocation](image)

To identify a device uniquely in the SoD zone, DeviceID is defined as follows.

```java
public class DeviceID {
    int endPointNumber; // endpoint of the device
    String ipAddress; // identify the device uniquely
}
```

The resource profile used internally is defined as follows.

```java
private class InternalDeviceProfile {
    String deviceState; // whether device is existed or not
    DeviceID deviceID;
    String deviceType; // keyboard, mouse or monitor
    String udi; // information field
    String deviceFile; // path to the device file
    String infoState; // current state of profile object
}
```

As computing peripherals, USB protocol devices are prevalent. The following example represents USB device about its information.

```java
public class USBDeviceInfo {
    int bus_number;
    int linux_device_number;
    double usb_speed;
    double usb_version;
    String linux_sysfs_path;
    int vendor_id;
    int product_id;
    String vendor_name;
    String product_name;
    String owner_id;
}
```

Table 2 shows the data format for a monitor, a mouse and a keyboard resource.

Data format for the monitor represents pixels of rectangle area. The real frame buffer data are following the monitor header described in Table 3.

Mouse data format indicates the relative coordinate of mouse point and status. Keyboard data format says the key code while key is pressed.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Type</th>
<th>Data</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitor</td>
<td>int</td>
<td>encoding_type</td>
<td>0 : raw, 1 : hextile</td>
</tr>
<tr>
<td></td>
<td>int</td>
<td>x</td>
<td>0 : data, -1 : change resolution</td>
</tr>
<tr>
<td></td>
<td>int</td>
<td>y</td>
<td>If x is 0, beginning y axis.</td>
</tr>
<tr>
<td></td>
<td>int</td>
<td>w</td>
<td>If x is -1, byte per pixel (4 or 2)</td>
</tr>
<tr>
<td></td>
<td>int</td>
<td>h</td>
<td>Pixel width (fixed)</td>
</tr>
<tr>
<td>Mouse</td>
<td>String</td>
<td>msg_type</td>
<td>Mouse command</td>
</tr>
<tr>
<td></td>
<td>int</td>
<td>dx</td>
<td>Relative x value</td>
</tr>
<tr>
<td></td>
<td>int</td>
<td>dy</td>
<td>Relative y value</td>
</tr>
<tr>
<td></td>
<td>int</td>
<td>dz</td>
<td>Wheel: 1(up), -1(down)</td>
</tr>
<tr>
<td></td>
<td>int</td>
<td>buttons_state</td>
<td>Button status</td>
</tr>
<tr>
<td>Keyboard</td>
<td>String</td>
<td>msg_type</td>
<td>Keyboard command</td>
</tr>
<tr>
<td></td>
<td>String</td>
<td>press</td>
<td>Whether key is pressed or not?</td>
</tr>
<tr>
<td></td>
<td>String</td>
<td>key_code</td>
<td>What’s the key code?</td>
</tr>
</tbody>
</table>
V. Prototype Implementation

This section describes the operation environment and the result of prototype implementation.

The test environment for operation consists of three parts:

- Management system: includes virtualization infrastructure, a management tool for an administrator and users
- Android SoD client: Froyo-powered Android phone as a mouse, a keyboard and a monitor function
- IOS SoD client: IOS-powered iPad as a mouse, a keyboard and a monitor function

![Figure 5. Operation screen shot of Android SoD client](image)

Figure 5 shows the screen shot of real SoD client operation.

The first figure describes that the selection check box for the phone to set which function of the phone is provided.

The second figure shows the screen of phone when the phone is selected as a monitor, keyboard and monitor.

The third figure shows management interfaces containing available virtual machines (top left), registered users (bottom left), available resources such as keyboards, mice, monitors, printers, USB storages (right side), and the currently operating virtual machines including attached user and resources.

Figure 6 shows the example operation combining with iPad SoD client and Android SoD client.

The administrator or users can select peripheral devices in the resource pool which is registry of virtualized peripheral devices. This example operation shows that the user select IOS SoD client (iPad) as a monitor and Android SoD client as a mouse and a keyboard. This operation can be done dynamically in runtime.
Figure 6. Example of combined operation

The result of prototype shows similar functions of VNC or RDP solutions but we can configure the resources dynamically whenever the user need to change his computing environment by using distributed available peripherals on demand. This is the main advantage points of this proposal.

VI. CONCLUSIONS

As the proliferation of desktop virtualization, we need more flexible I/O options such as smartphone as a peripheral of user’s virtual desktop.

Due to this needs, we propose Android-based SoD (System on-Demand) client for remote presentation in virtual desktop environment.

Each I/O resources of SoD client can be separable and can be allocated to the virtual desktop dynamically. This function accelerates the usage of smartphone and provides more flexible I/O selection option to users.

The challenge point of this paper is separating and virtualizing traditional peripherals of desktop. After this, we can redirect each virtualized I/O to any SoD client device on demand.

This scheme may contribute to realizing VPI (virtual peripheral interface) just targeting for a smartphone. Therefore, we need to allow for following issues as the future research works to evolve into a full-fledged system. Firstly, we allow for various smartphones and make them as SoD client. Secondly, we also need to deal with performance issues while a resource connection through IP network.

ACKNOWLEDGMENT

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].

REFERENCES