A Research of USB Device Redirection Mechanism over IP network in Desktop Cloud System

Yi-min ZHOU
Dept. of Computer
School of Optical-Electrical and Computer Engineering,
University of Shanghai for Science and Technology
Shanghai, China, 13901854513

Hui-hui GUO
Dept. of Computer
School of Optical-Electrical and Computer Engineering,
University of Shanghai for Science and Technology
Shanghai, China, 18801929086

***Abstract*—As most existing device sharing technologies are only abstract operations sharable, such as read and write, virtual desktop users can't share the device-specific characteristics of USB devices attached to cloud terminal. In this paper, we propose USB device redirection as a peripheral bus extension over an Internet Protocol (IP) network. Cloud terminal extends the local peripheral bus to a virtual desktop (VD). The main component of this extension is a virtual peripheral bus driver, which resides in the lowest layer in an operating system. Our experiments show that this approach enables virtual desktop applications to access diverse features of all USB devices and has sufficient I/O performance.**

Keyword-desktop cloud; virtual desktop; USB device redirection; IP Network

I. INTRODUCTION

With the rapid development of computing technology and high-speed internet technology, more and more enterprises commit to implement high resource utilization and data security. Conventional work environment that comprise multiple computers connected via a local area network (LAN) is unable to meet enterprise users' demands. Cloud computing is a new way of internet resource utilization [1]. Resources provided via internet are virtualized and can be dynamic expanded, which greatly improves the utilization of enterprise resource and data security.

Desktop cloud system is a private cloud [2] and it has taken significant changes to traditional desktop management. It allows cloud terminal to access cross-platform applications and the whole virtual desktop through a thin-client or any devices that connected to network [3]. Universal Serial Bus (USB) [4] is one of the most sophisticated peripheral interfaces, providing serialize I/O, hot-plug and universal connectivity. It has become a general interface for most personal computer and intelligent devices.

Many existing device sharing technologies that based on LAN have been proposed. However, these technologies are implemented through the sharing of abstract functions that reside in a high-level of operating system, such as read and write. Some low-level features, such as format and divide USB storage device operations, can't be shared, so virtual desktop can't access these new features of cloud terminal USB devices.

In this paper, we propose USB device redirection as a peripheral bus extension over an Internet Protocol (IP) network. Cloud terminal expands the local peripheral bus to a virtual desktop. We defined the expanded peripheral bus driver as virtual peripheral bus driver and it resides in the lowest layer of operating system. Using a virtual peripheral bus driver, virtual desktop can accesses a diverse range of terminal USB devices over networks.

II. RESEARCH BACKGROUND AND TARGET

A. *Research Background*

An operating system is used to manage various resources on a computer. It also provides some generalized interfaces for applications (e.g., read and write). However, most traditional device sharing mechanisms only allow applications to access remote devices through these abstract functions [5]. Computers could not share any more lower-layer operations.

Take a storage device as an example in desktop cloud system. When a hard disk is shared by conventional sharing mechanism, such as samba [6] or NFS, virtual desktop can't deal with this device directly. It controls this device with file operations provided by remote procedure calls which are similar to system calls and works fine for storing and retrieving data. However, with the constantly emerging of new-featured USB devices, conventional device sharing mechanisms could not share these new functions for the following reasons:

First, it is the upper-layer abstract functions that sharable, the more fine-grained and device-specific operations are not supported. For example, in LINUX operating system, NFS is implemented using virtual file system (VFS). As VFS resides in an upper-level layer and VFS protocol does not define lower-level methods, such as format or eject a remote storage device, virtual desktop unable to share these I/O operations when using conventional sharing mechanisms.

Second, since the complex differences between operating systems, to achieve a high degree of interoperability is difficult. Many conventional sharing mechanisms only support the same operating system because they extend abstraction layer for sharing. As different operating system may have different abstraction layer and it always causes bad interoperability.

Considering the deficiency of conventional device sharing mechanism, it is urgently for us to develop a new device sharing mechanism, which providing device sharing for the

This project is financially supported by Innovation Program of Shanghai Municipal Education Commission (11YZ121).

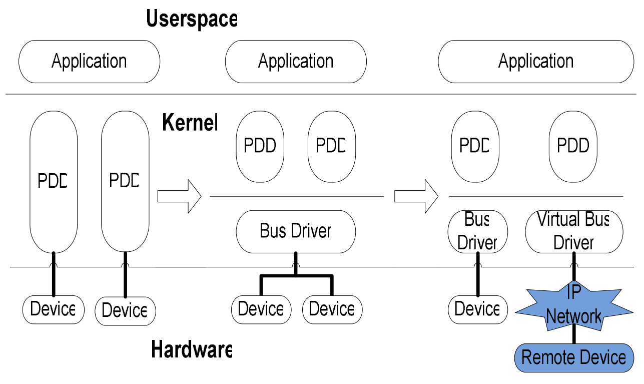
new-featured USB devices as they are released and possessing a high interoperability.

B. *Research Target*

In traditional operating system, the device driver is used to communicate with the local physical device and provides an interface to allow user-land applications to access device functionality. This mechanism leads to the drawbacks mentioned above.

Nowadays, the improvement of computer hardware makes intelligent peripheral buses (e.g., USB) commonplace. A device driver is usually separated into two parts: a bus driver manipulating peripheral interfaces and a per-device driver controlling devices on the peripheral interfaces. Device drivers are generally responsible for data transmission with the local attached device, while the operating system provides dynamic device configuration.

In this paper, USB device redirection approach extends the peripheral bus over an IP network by using a virtual bus driver. The virtual bus driver provides an interface to remote redirected device by encapsulating peripheral bus request commands in IP packets and transmitting it through the LAN. Figure 1 shows the evolution of device driver architecture.



(Stage: conventional (left), current (middle) and virtual (right) respectively)

Figure1. Evolution of Device Driver Architecture

USB device redirection approach fully utilizes existing dynamic device management mechanism and resolves the drawbacks of traditional approaches for the following reasons:

First, redirected device is functionally and diversely available. Since redirection operations are implemented in the lowest layer of an operating system, the bus driver layer conceals only the bus differences and does not affect the per-device operations. Meanwhile, a variety of USB devices can be redirected in this way, because a virtual bus driver is independent of the per-device driver, and it supports all the devices on the peripheral interfaces.

Second, there is no need to modify either existing operating system or applications to access remote redirected devices. As the virtual bus driver conceals the implementation details of network redirection mechanism and device drivers control remote devices through the virtual bus driver, components of the operating system and applications do not notice any difference between the access interfaces of redirected devices and locally-attached ones.

III. USB Device Redirection

In this section, we first discuss USB device driver model, and then detailed describe the design and implementation of USB device redirection.

A. *USB Device Driver Model*

Figure 2 shows each layer of USB device driver. The granularity of operations on the lower layer is fine-grained. Data size and temporal restrictions of each operation are smaller than that in the upper layer.

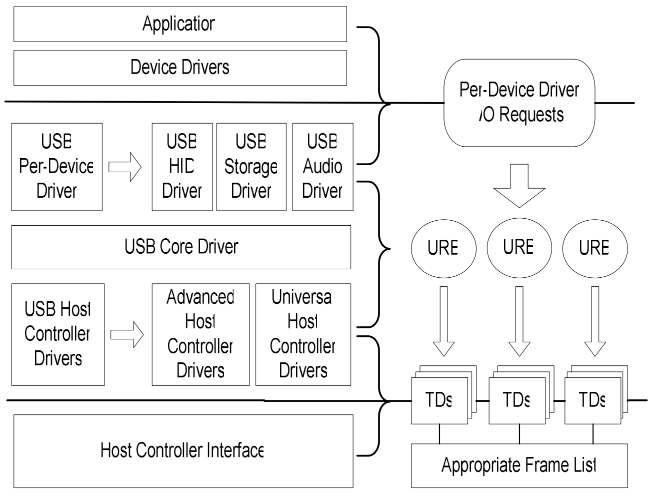


Figure 2. USB Device Driver Model

A USB Per-Device Driver (USB PDD) is used to control individual USB device. When applications or other device drivers request I/O operations to a USB device, USB PDD converts I/O requests to a series of USB commands and then transmits them to USB Core Driver in the form of USB Request Blocks (URBs). A USB PDD only uses a device address, an endpoint address, an I/O buffer and some additional information required for device communication.

USB Core driver is responsible for the dynamic configuration and management of USB devices. When a new USB device is attached to the bus, USB Core Driver enumerates it and then loads an appropriate USB PDD for this device. USB Core Driver also provides a set of interfaces to the upper USB PDDs and lower USB Host Controller Drivers.

A USB Host Controller Driver (USB HCD) is used to manage the data transmission between host computer and USB devices. USB HCD receives URBs from USB Core Driver and then divides them into smaller requests, named Transfer Descriptors (TDs). TDs are scheduled by their transfer types and are linked to appropriate frame lists in HCD for delivery to manipulate USB devices.

B. *Design and Implementation*

Figure 3 shows the implementation of USB device redirection in desktop cloud system. We add a Virtual Host Controller Interface driver (VHCI) as a virtual bus driver. The VHCI acts as a USB HCD and it converts a URB into a request block and sends it to remote redirected USB device. A Stub driver in cloud terminal is also added as a new type of USB PDD. It is used to decode received packets from remote machine, extract the URBs, and then submit them to the local USB device.

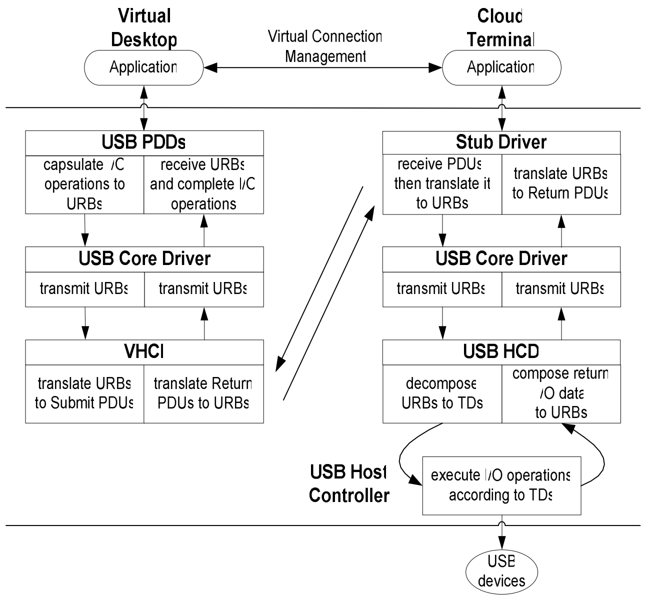

Virtual Desktop
Application
Virtual Connection Management
Cloud Terminal
Application
USB PDDs
capsulate I/O operations to URBs
receive URBs and complete I/O operations
USB Core Driver
transmit URBs
transmit URBs
VHCI
translate URBs to Submit PDUs
translate Return PDUs to URBs
Stub Driver
receive PDUs then translate it to URBs
translate URBs to Return PDUs
USB Core Driver
transmit URBs
transmit URBs
USB HCD
decompose URBs to TDs
compose return I/O data to URBs
USB Host Controller
execute I/O operations according to TDs
USB devices

Figure 3. Design and Implementation of USB Device Redirection

When cloud terminal that attaching a USB device is connected to a virtual desktop via IP network, the VHCI driver informs USB Core Driver of port status change. USB Core Driver maps a USB PDD according to received USB device information, and then loads an appropriate USB device driver.

When virtual desktop applications access to mapped device, USB PDD calls usb_submit_urb(struct *urb, ..) to encapsulate I/O requests into URBs and submit them to a lower layer. The VHCI driver translates a URB into a SUBMIT PDU (Protocol Data Unit) (shown in Table Ⅰ) and transmits it to the Stub driver by calling vhci_tx_loop(struct *urb, ..).

TABLE I. SUBMIT AND RETURN PDU (PROTOCOL DATA UNIT)

Byte	SUBMIT	RETURN
0-3	SUBMIT	RETURN
4-7 8-11 12-15	bus number device number sequence number	
16-19	IN/OUT & I/O type & endpoint	reserved
20-23 24-27 28-31	transfer flags buffer length number of included transaction	
32-35 36-39	transaction interval control buffer	error count transfer status
40-	isochronous descriptors (if available)	
	I/O buffer (if available)	

The Stub driver receives the SUBMIT PDU via stub_rx_loop(struct *urb, ..), creates a new URB from it, and then submits the URB to a real USB host controller by usb_submit_urb(struct *urb, ..). USB HCD translates incoming URBs to a series of TDs corresponded to the actual USB processing frame, which will control the host controller chip to finish I/O operations of USB device.

When I/O operations completed, USB HCD transmits return URBs to the Stub driver through USB Core Driver. The Stub Driver calls stub_tx_loop(struct *urb, ..) to set up a RETURN PDU (shown in Table Ⅰ), which includes the status of I/O and input data if available, and then transmit it through the IP network. The VHCI driver receives a RETURN PDU and translates it to URBs via vhci_rx_loop(struct *urb, ..), and finally transmits it to USB PDD through USB Core Driver. Table Ⅱ shows functions mentioned above.

TABLE II. USB DEVICE REDIRECTION FUNCTIONS

```
void vhci_tx_loop(struct urb *urb, ..) {
      struct pdu *pdu;
      while (1) {
            if(vhci_urb_enqueue(pdu, urb)<0)       break;
            if (vhci_send_pdu_submit(pdu) < 0)      break;
            if (vhci_send_pdu_unlink(pdu) < 0)      break;
            wait_event_interruptible(pdu->waitq_tx,
            (!list_empty(&pdu->priv_tx)||!list_empty(&pdu->unlink_tx))); }
}
```

```
void stub_rx_loop(struct urb *urb, ..) {
      struct pdu *pdu;
      while (1) {
            if (usbip_event_happened(pdu))       break;
            stub_rx_pdu(pdu);
            if(stub_urb_dequeue(urb, pdu)<0)     break; }
}
```

```
void stub_tx_loop(struct urb *urb, ..) {
      struct pdu *pdu;
      while (1) {
            if (usbip_event_happened(urb))       break;
            if(stub_urb_enqueue(pdu, urb)<0)     break;
            if (stub_send_ret_submit(pdu) < 0)   break;
            if (stub_send_ret_unlink(pdu) < 0)   break;
            wait_event_interruptible(pdu->waitq_tx,
            (!list_empty(&pdu->priv_tx)||!list_empty(&pdu->unlink_tx))); }
}
```

```
void vhci_rx_loop(struct urb *urb, ..) {
      struct pdu *pdu;
      while (1) {
            if (usbip_event_happened(pdu))      break;
            vhci_rx_pdu(pdu);
            if(vhci_urb_dequeue(urb, pdu)<0)   break; }
}
```

The desktop cloud system transfers all PDUs by a TCP/IP connection, which is established by user-land software. Socket descriptor passed to the VHCI and Stub driver is implemented via /proc file system. The reason why we do not use UDP/IP protocol is that the characteristics of the transmission errors of USB and UDP/IP are quite different.

IV. EVALUATION AND CONCLUSION

A. *Evaluation*

In this section, we will show performances and characteristics of USB device redirection approach. A desktop cloud system is usually based on Virtual Desktop Infrastructure (VDI) solutions [7]. The foundation of virtual desktop solutions is server virtualization. Hypervisor is the core of the most popular virtualization technologies. We can use it to create a customized virtual desktop (VD). Administrator in cloud side deploys the operating system and a variety of applications for each user. Cloud terminal is connected to virtual desktop through the IP network, and then visits the virtual desktop via desktop displays protocol.

TABLE III. EXPERIMENTAL MACHINE INFORMATION

Type	Cloud Terminal	VDs in Cloud Side
CPU	Samsung S5PV210 (ARMCortex-A8)	QEMU Virtual CPU (cpu64-rhel6)
Clock	600MHz-1GHz	2.26GHz
Memory	312M	1G
Operating System	Debian6.0.4 (linux-2.6.35.7)	Ubuntu11.04(linux-3.0.0-16) Windows XP SP3 Windows 7 Ultimate SP1

TABLE IV. EXPERIMENT RESULTS

Device Information	Virtual Desktop in Cloud Side					
	Linux		Windows XP		Windows 7	
U-Disk Data Size	**Read and Write Speed (MB/s)**					
	Read	Write	Read	Write	Read	Write
10M	3.107	2.875	2.728	2.457	2.835	2.594
50M	2.665	2.491	2.564	2.347	2.474	2.378
100M	2.653	2.488	2.471	2.354	2.476	2.367
500M	2.493	2.407	2.378	2.212	2.439	2.198
1G	2.458	2.386	2.382	2.079	2.382	2.127
5G	2.336	2.277	2.263	2.106	2.338	2.083
Printer	**I/O Operations Performance**					
Printing	OK		OK		OK	
Copying	OK		OK		OK	
Scanning	OK		OK		OK	
Sound Card	**I/O Operations Performance**					
Audio In	OK		OK		OK	
Audio Out	OK		OK		OK	

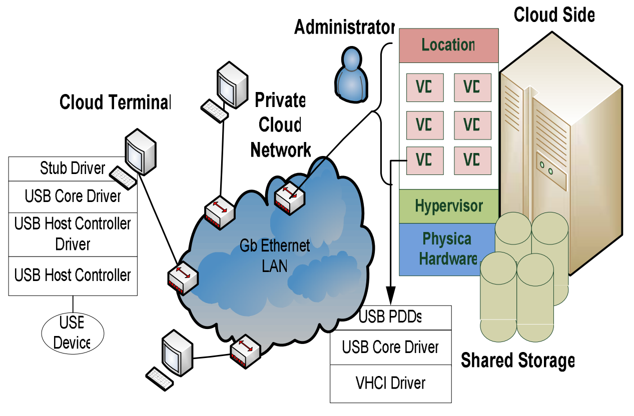


Figure 4. Architecture of Desktop Cloud System

Figure 4 builds the experimental environment for USB device redirection approach. Table Ⅲ shows machine information of cloud terminal and virtual desktop. Cloud terminal visits virtual desktop via spice [8], an open source desktop displays protocol. USB Devices to be evaluated include three types: storage, control and audio. We choose Teclast Coolflash USB3.0 U-disk as storage device, capacity for 8G and USB2.0 compatible. The control device we select is a Samsung SCS-4x20 series PCL6 multi-function printer. Audio device is a sound card, 7.1 Channel Sound. Table Ⅳ shows the evaluation results.

B. *Results and Conclusion*

By the experiments, we show that it is reasonable to expand the peripheral bus over an IP network to implement USB device redirection in desktop cloud system. We can functionally use various remote USB devices and different operating systems can access remote redirected devices without any modification. Furthermore, the I/O performance of remote USB devices in LAN is sufficient for actual usage.

As far as USB device redirection mechanism works properly, it still has the following drawbacks. First, since device redirection is implemented in a lower layer of an operating system and the raw device functions are being shared, it is unable to provide concurrent access to a remote USB device. Second, as the proposed approach is sensitive to IP network, issues such as network delay and jitters will influence usage of remote redirected devices. Third, USB device redirection mechanism has only been widely applied in LAN environment at present.

In future work, with the application of network lock mechanism, concurrent access to a remote redirected device will be realized. Meanwhile, the rapid progress of networking technologies will alleviate problems of network delay and jitters. Besides, we will continue to improve the USB device technology to support various network environments efficiently, such as a wireless network and a wide area network (WAN).

REFERENCES

[1]. Q. Zhang, L. Cheng, and R. Boutaba, Cloud computing: state-of-the-art and research challenges, Journal of Internet Services and Applications (JISA), Vol. 1, No. 1, pp. 7-18, February 2010.

[2]. Xinyu Miao; Jing Han; The Design of a Private Cloud Infrastructure Based on XEN, 2011 10th International Symposium on Distributed Computing and Applications to Business, Engineering and Science.

[3]. A. Berryman, P. Calyam, A. Lai, M. Honigford, VD Bench: A Benchmarking Toolkit for Thin-client based Virtual desktop Environments, Proc. of IEEE CloudCom (2010).

[4]. Universal Serial Bus Revision 2.0 specification. http://www.usb.org/developers/docs/.

[5]. Wonhong Kwon, Han Wook Cho, and Yong Ho Song; Design and Implementation of Peripheral Sharing Mechanism on Pervasive Computing with Heterogeneous Environment, College of Information and Communications, Hanyang University, Seoul, Korea (2007).

[6]. SAMBA, http://us1.samba.org/samba/.

[7]. Prasad Calyam, Rohit Patali, Alex Berryman, Albert M, Lai, Rajiv Ramnath; Utility-directed resource allocation in virtual desktop clouds, Computer Networks 55 (2011) 4112–4130.

[8]. SPICE, http://spice-space.org/.