vMobiDesk: Desktop Virtualization for Mobile Operating System

Kui Su, Pengfei Jiang, Zonghui Wang, Wenzhi Chen
College of Computer Science
Zhejiang University, Hangzhou, 310027, China
Email: sukuias12@zju.edu.cn

Abstract—**With the rapid adoption of smartphones and tablets, virtual desktop infrastructure (VDI) for mobile devices emerges as one of the key concept in mobile cloud computing (MCC). However, existing VDI products are originally designed for personal computer operating systems (e.g. Windows, Mac OS, Ubuntu, etc.). The user experience is greatly degraded while viewing and operating a PC desktop from a smartphone or a tablet. In this paper, we propose techniques to virtualize the desktop of mobile operating systems (e.g. Android, iOS, Winphone, etc.) for mobile users and we have implemented a prototype system on Android which is one of the most popular mobile operating systems. The experimental results show that our system enables mobile users to achieve almost the same experience as in the local while viewing and operating the remote mobile desktops.**

Keywords—Mobile Operating System; Virtual Desktop; Remote Computing; Virtualization

I. INTRODUCTION

As an emerging trend, virtualization [1, 2, 3] has been widely used in cloud computing [4, 5, 6] over the past decade. Among those virtualization applications, desktop virtualization has become an important branch [7, 8, 9]. In desktop virtualization environment, all applications and operating system codes are executed in a server which lies in a remote data center. End user only needs a thin client which handles display, keyboard and mouse combined with adequate processing power for graphical rendering and network communication. The client no longer has to keep user state and communicate with the server by using a remote protocol. The protocol allows graphical displays to be virtualized, and transmits user input from the client to the server [10]. Many productive desktop virtualization systems have been developed and applied to various commercial applications since they provide a lot of advantages for IT enterprises such as reducing maintenance and operating costs and improving resource utilization efficiency.

With the rapid adoption of smartphones and tablets, virtual Desktop Infrastructure (VDI) for mobile devices emerges as one of the key concept in Mobile Cloud Computing (MCC) [11]. At the same time, the increasingly popular BYOD [12,13,14] policy has enabled more and more IT enterprises to encourage employees accessing VDIs with their own devices (the most people prefer smartphones and tablets due to convenience) anytime and anywhere, which greatly reduces IT costs and improves productivity.

However, existing VDI products are originally designed for personal computer operating systems (e.g. Windows, Mac OS, Ubuntu, etc.). They can only provide mobile users with PC desktops which are not perfectly compatible with mobile devices. The user experience is greatly degraded while accessing and operating a PC desktop from a smartphone or a tablet. In this paper, we introduce vMobiDesk, a system to virtualize the desktop of mobile operating systems (Android, iOS, windows phone, etc.) for mobile users. We have implemented a prototype system of vMobiDesk on Android which is one of the most popular mobile operating systems. The experimental results show that our system enables mobile users to achieve almost the same experience as in the local while accessing and operating the remote mobile desktops.

This paper presents the design and implementation of vMobiDesk. Section 2 presents the motivation of vMobiDesk system in detail. Section 3 describes the overall architecture and implementation of vMobiDesk system. Section 4 presents experimental results measuring vMobiDesk system performance. Section 5 discusses related work. Finally, we present some concluding remarks.

II. MOTIVATION

In this section, we introduce the motivation of vMobiDesk system which performs better than traditional VDI products for mobile devices and brings many important benefits to mobile cloud computing.

Firstly, traditional VDI products only provide PC desktops, which leads to bad user experience for mobile users. Because the screen size of mobile devices is so small that it cannot well display the whole PC desktop. what is worse, applications running on PC operating systems are often designed for 13 inch or bigger screen, which makes it difficult to use these applications in 5 inch screen of mobile devices.

Secondly, in vMobiDesk system, the mobile operating systems are running on virtual machines, mobile users can suspend, hung up, delete and stop the virtual desktop through the hypervisor anytime and anywhere. Most mobile devices are not equipped with the above functions.

Furthermore, there are a variety of mobile devices such as Apple, Samsung, Huawei, etc. and many mobile operating systems such as iOS, Android, Winphone, etc. Nowadays. For one application, programmers have to develop it multiple times for different mobile OS and devices. But in vMobiDesk system, one mobile OS desktop can run on all of the mobile devices. Apps programmers don't need to do cross-platform development any more. Of course, the vMobiDesk client is necessary for each mobile platform.

III. DESIGN AND IMPLEMENTATION

In this paper, we propose vMobiDesk, a prototype system which provides mobile users with remote access to virtual mobile desktop such as Android desktop. In vMobiDesk, mobile users can experience a virtual Android desktop as the same as in the local. In this section, we first introduce the overall architecture of vMobiDesk system, then we describe the core modules of vMobiDesk system in detail, respectively.

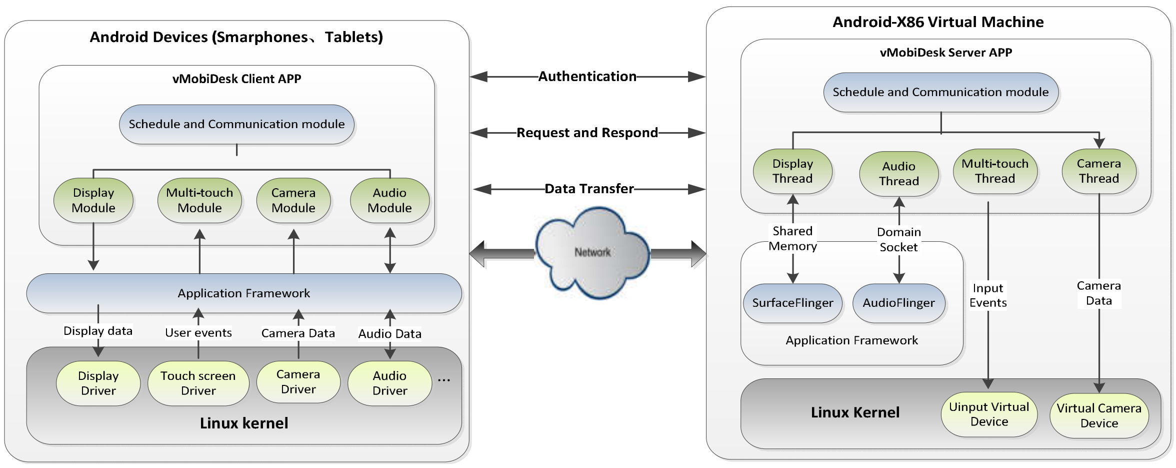

Fig. 1. The system architecture of vMobiDesk

A. *Overview of System Architecture*

vMobiDesk is architected as a Client-Server system. It is implemented on Android system which is one of the most popular mobile operating system. As shown in figure 1, the vMobiDesk client is an application running on Android devices (smartphones or tablets). The client is composed of four parts. The display module is designed to display the virtual android desktop on the mobile devices. The multi-touch module mainly handle the mobile users' input events. The camera and audio modules are designed to provide mobile users with good experience of multimedia. Meanwhile, the server of vMobiDesk which is running on an Android-x86 virtual machine provides the corresponding services to the mobile client. The client is connected to the server through the remote access protocol with security authentication.

B. *Display Virtualization*

To make vMobiDesk a viable replacement to the traditional desktop computing model, it needs to be able to deliver the look and feel of all unmodified desktop applications end-users expect. vMobiDesk must work within the framework of existing display systems, intercepting display content from unmodified applications and redirecting these content to remote clients. In order to provide good performance, the virtualization must intercept display content at an appropriate abstraction layer so that it provide sufficient information to optimize the processing of display content in a latency sensitive manner. Furthermore, to support transparent user mobility and eliminate client administration complexity, vMobiDesk should support the use of thin, stateless clients, by ensuring that all persistent display state is stored in the server infrastructure.

There are many remote display protocols for mobile VDI system, such as VNC [15], RDP [16], SPICE [17], etc. However, only VNC can be utilized as an Android server. Because VNC is platform-independent – there are clients and servers for many GUI-based operating systems. The primary drawback of VNC for vMobiDesk server is that it mainly read pixel data from framebuffer. The framebuffer is only updated after the system executes drawing functions, saves data to buffer of graphic card and display on the screen. Therefore, the reading at framebuffer layer takes more delay and the server must consume much CPU to display the pixel data on server's screen. What is worse, as the increasing number of mobile VDI users, the overall performance of the mobile VDI system will be greatly degraded.

vMobiDesk does not intercept the display content at framebuffer layer because it consumes much server's CPU and causes respond delay. vMobileDesk intercepts the display content of Android virtual desktop at the system service layer. Surfaceflinger is a core service for Android display system, it is mainly responsible for composing all the surfaces from the applications, then generates a complete desktop surface and write it to the hardware framebuffer for displaying on the screen. vMobiDesk intercept the complete desktop content before it is written to the framebuffer. For security and efficiency, the intercepted content is temporarily written to a shared memory region, then the display thread of vMobiDesk reads it and transmits the content to the client's display module through the network. On the server side, because it is unnecessary to display the desktop on server's screen, the desktop content will not be written to the hardware framebuffer. The above design of display solution for vMobiDesk can significantly reduce server's CPU consumption and improve the respond speed, which provides mobile users with good visual experience for virtual mobile desktops.

C. Input Redirection

With the advent of smartphones and tablets, interactions through keyboard and mouse are no longer relevant as majority of smartphones are equipped with multi-touch screens. However, there is no remote computing protocol specifically designed for touch screen input devices. Existing VNC smartphone client and server applications are designed assuming the other end of communication is a desktop that can be controlled through keyboard and mouse operations. However, vMobiDesk provide mobile users with remote access to an Android desktop, the touch input events from mobile users should be seamlessly redirected to the server.

In vMobiDesk, touch input events are loosely classified into two types: single-touch events and multi-touch events. On the client side, user's touch input events are intercepted at the device driver layer to ensure that the input events are original and application-independent. This is because that the server and client may be running Android system of different versions, but at the driver layer, the input events are consistent. If it is a single-touch event, the client just forwards its coordinate to the server and re-execute it. But if it is a multi-touch event, for example, mobile users prefer to use multiple fingers to scale up and down the web pages on the screen, there will be three steps. First, the event is identified as a multi-touch event. The multiple coordinates are combined into a new event. Then this event is forward to the server. Lastly, the event will be injected into the input system. In practice, because the server is running Android-x86 system on a virtual machine, this is no real touch screen device. A virtual device must be created to receive the multi-touch events and execute it.

D. Audio Support

For mobile users, listening to popular music and making voice communications with family or friends are indispensable parts in life. To make it widely accepted by mobile users, mobile VDI system must provide users with good experience for audio redirection.

In vMobiDesk, when an user play a music in the virtual mobile desktop, the audio data will be intercepted by server's audio thread and then transmitted to the client in real time. On the client, audio data is received and replayed by audio module. In practice, to reduce server's CPU usage, audio data is prevented to be written in server's audio device, which means that we are unable to hear the music sound on the server side. For voice communication, the way is to the contrary, user's voice is captured and transmitted to the server's voice communication applications for post-processing. In order to ensure real-time, audio data is buffered on both the server and client sides. In our experiments, mobile users can listening to music and make voice communications in their virtual mobile desktops in almost real-time.

E. Remoting Camera

Camera is one of the most important modules in smartphones or tablets. With a camera, mobile users can take photos and record videos anytime and anywhere to record and share their lives with family and friends. In order to provide good experience for mobile users in mobile VDI system, it is necessary to implement camera redirection so that mobile users can use the camera in the virtual mobile desktop as the same as in the local.

In vMobiDesk, camera redirection is designed as a core module. A mobile user's any action on camera application in the virtual mobile desktop will be redirected to the mobile device's local camera in real-time. For example, when an user opens a camera application in a virtual mobile desktop, it is detected by server's camera thread immediately. Then the camera thread delivers the message to the client's camera module to open the mobile device's local camera. In this way, mobile users can use their cameras as in the local. To completely implement the camera redirection, virtual camera devices must be created on vMobiDesk's server side. The number of camera devices may be one or more, which is determined by the camera number of user's mobile device. The virtual camera devices are created to receive and process the camera data from the client's physical cameras, then return the results to the camera applications.

In camera redirection, the biggest challenge is how to handle the recording videos which often carry a large amount of video data. Generally, recording video data is transmitted to the server's virtual camera devices by the client. After handled on the server, the video must be displayed on the client's screen. In the above processes, video data are transmitted two times in total, but due to the limited mobile network bandwidth in mobile VDI system, the transfer of recording video data will cause high delay and terrible experience for camera application such as poor video quality and dropped frames. To address the above problems, we propose to only do the first transfer which is for processing the camera data on the server, the second transfer is prevented because it is just for displaying or previewing on the client, its local camera data can be utilized for displaying or previewing. By this means, the vMobiDeskt system can definitely reduce delay and improve user experience for camera applications in mobile VDI system.

F. Remote File System

Nowadays, cloud storage is widely used, it is a model of data storage in which the digital data is stored in logical pools, the physical storage spans multiple servers (and often locations), and the physical environment is typically owned and managed by a hosting company such as Smugmug, Dropbox, Synaptop and Pinterest. People and organizations buy or lease storage capacity from the providers to store user, organization, or application data. Therefore, it is indispensable to provide a remote file system for mobile users to upload and download resources conveniently in a mobile VDI system.

In vMobiDesk, a samba [22] server is running as a thread in the client application and a samba client is running on the server side. While the client is connected to the server, the samba client also connects to the samba server immediately. Then the samba client will mount the mobile device's SD card on the server. After that, the user is able to view the content of the SD card in the virtual mobile desktop. At anytime and anywhere, mobile users can upload the local files such as photos and videos to the server through the file manager in the virtual desktop. In addition, mobile users can also download movies or documents in the virtual desktops and then copy them to the SD cards of their mobile devices.

IV. Performance Evaluation

We have implemented a prototype vMobiDesk system for serving Android desktop computing environments. In this section, we present the performance evaluation of our implementation of vMobiDesk with a series of experiments.

In our experimental setup, the server machine which hosts Xen 4.2 hypervisor has a 2.66 GHz Intel Core i7 − 920 processor and 8 GB of RAM. The server system Android-x86-5.1.1 is running in a virtual machine created by Xen hypervisor. The client machine is a 16GB Nexus 9 tablet which runs Andorid-5.1.1 system. We use a 100 Mbps, 1ms latency LAN network to build local Wifi environment for the client to connect to the server. We mainly evaluate the performance of vMobiDesk in terms of respond time, network bandwidth consumption and system overhead.

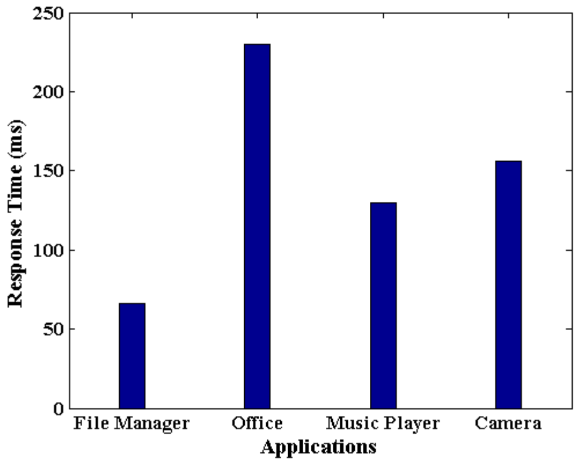

Fig. 2. The response time of running different applications in vMobiDesk system.

A. Response Time

In vMobiDesk, the server provides mobile users with remote access to the virtual mobile desktop through the local Wifi. Respond time is one of the most important metric for users' experience on applications such as File Manager, Office software (e.g. Word, Excel, PowerPoint), Music Player and Camera application. We have measured the average response time for different applications in vMobiDesk. For example, the average response time of Office softwares are computed from the response time of opening a Word document, typing several words, inserting a picture, etc..

As shown in figure 2, all the applications take less than one second while running in the virtual mobile desktop. The File Manager performs the best because it only contains simple operations such as opening and closing a file. The Office contains much complex operations, for example inserting an animation to the PowerPoint, which is CPU and memory intensive. The Music Player takes only 150 milliseconds because we have buffered audio data both on the server and the client side. The Camera also performs well, users can take photos and record videos as the same as in the local. Though the recording may produce large amounts of video data, our optimization of reducing the twice transfer of video data to only once makes it still clear and fluent.

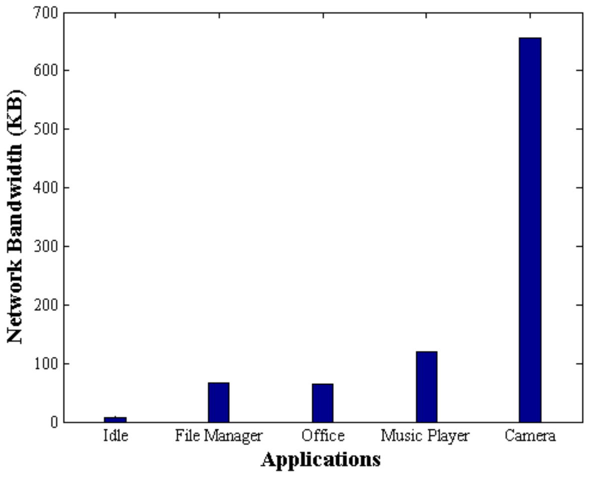

Fig.3. The bandwidth consumption of running different applications in vMobiDesk system.

B. Bandwidth Consumption

This experiment is designed to measure the detailed bandwidth consumption of each application in vMobiDesk system. The network environment is a 100 Mbps LAN Wifi. As shown in figure 3, the File Manage and Office consume almost the same bandwidth because they both produce only static display data. The Music Player consumes more bandwidth because the music contains audio data which is complicated. The Camera takes up the most bandwidth since it refers to redirecting recording video data. Fortunately, we have optimized vMobiDesk to only transfer the video data one time, the bandwidth consumption is greatly reduced.

C. System Overhead

We now present a CPU and memory usage profiling of vMobiDesk. We collect the CPU and memory statistics

TABLE 1: THE OVERHEAD OF THE vMOBIDESK CLIENT

Overhead / Application	CPU usage (%)	Memory usage (MB)
Idle	5	20
File Manager	8	22
Office	12	45
Music Player	11	60
Camera	15	177

while executing the server and the client application of vMobiDesk system. As shown in table 1, on the client side, the idle state consumes about 5% CPU and 20MB memory. While a mobile user is launching different applications in the virtual mobile desktop, the CPU usage rises to about 8%, 12%, 11% and 15%, accordingly. The memory usage also rises to 22MB, 45MB, 60Mb and 177MB. The table 2 gives the CPU and memory consumption on of vMobDesk on the server side. Because vMobiDesk has modified the Android-x86's system services such as Surfaceflinger service, Media Server, and even the Linux kernel to support desktop virtualization, the most of the overhead is the data transfer from Android system to the server application.

TABLE 2: THE OVERHEAD OF THE vMOBIDESK SERVER

Overhead / Application	CPU usage (%)	Memory usage (MB)
Idle	4	13
File Manager	10	24
Office	14	58
Music Player	15	66
Camera	19	194

As described in table 2, when it is idle, vMobiDesk only consume 4% CPU and very little memory. Even running different applications, the CPU usage is always under 20% and the memory usage is less than 200MB. In general, vMobiDesk takes up less resource and provide good performance.

V. RELATED WORK

Many productive desktop virtualization systems have been developed and applied to various commercial applications since they provide a lot of advantages for IT enterprises such as reducing maintenance and operating costs and improving resource utilization efficiency. VNC [15] and THINC [18] are famous thin-client systems proposed in academic research while in industry there are Microsoft Remote Desktop [16], Citrix XenDesktop [19], VMware View [18], Sun Ray and HP Remote Graphics and so on.

VNC (Virtual Network Computing) is a popular remote display system with RFB protocol. It uses a virtual driver to maintain local copy of the framebuffer state used to refresh its display and forward user input directly to the server. VNC provides a good performance for office applications but does not support multi-touch, audio and camera. Therefore it is not suitable for mobile VDI systems.

THINC and its portable version pTHINC intercepts lowlevel video driver commands and adopts a push mode to interact with client. Its codec is efficient for UI compression but suffers from compression performance degradation over multimedia content encoding. As a result, it can achieve a great multimedia playback performance with sufficient bandwidth but not for network environments with low bandwidth.

RDP (Remote Desktop Protocol) is widely used in desktop virtualization products such as Microsoft RDS and VMvare view. For office applications, such as a text editor or a spread-sheet, RDP is highly optimized and the display changes are quite small and have a sufficiently low frequency to cope with. However, it is proprietary and only supports Windows operating system. For mobile VDI system, RDP protocol is not a good choice.

MobiDesk [20] proposes a thin client solution for mobile devices by optimizing the WAN traffic involved in performing remote computing. The solution is primarily meant for mobile laptops and is similar in principle to other remote computing approaches. PCoIP [21] is another product that optimizes remote computing traffic, especially video over IP networks.

However, all these solutions assume the server is a PC (Windows 7, Ubuntu, Mac OS, etc.) desktop, as discussed in section 2, they don't address the problems of the bad user experience while displaying and operating a PC desktop from a mobile device such as smartphone or tablet.

VI. CONCLUSION AND FUTURE WORK

With the rapid adoption of smartphones and tablets, virtual desktop infrastructure (VDI) for mobile devices emerges as one of the key concept in mobile cloud computing (MCC). However, existing VDI products are originally designed for personal computer operating systems (e.g. Windows, Mac OS, Ubuntu, etc.). The user experience is greatly degraded while displaying and operating a PC desktop from a smartphone or a tablet. In this paper, we introduce vMobiDesk which provides mobile users with good experience on remote access of mobile OS desktop. We have implemented a prototype system on Android which is one of the most popular mobile operating systems. The experimental results show that our system enables mobile users to achieve almost the same experience as in the local while accessing and operating the remote mobile desktops.

However, though vMobiDesk performs well while running several applications such as File Manager, Office, Music Play and Camera Apps, it is still a experimental system. There are many other important applications such as Video Player and 3D games should be supported better. Furthermore, in the future work, we plan to utilize container techniques to virtualize the operating system on the server in order to improve the overall performance of vMobiDesk system.

ACKNOWLEDGMENT

This work is supported by the National Science and Technology Major Project of the Ministry of Science and Technology of China under grant 2013ZX03003010-002.

REFERENCES

[1] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. ACM SIGOPS Operating Systems Review, 37(5):164–177, 2003.

[2] Rich Uhlig, Gil Neiger, Dion Rodgers, Amy L Santoni, Fernando CM Martins, Andrew V Anderson, Steven M Bennett, Alain Kagi, Felix H Leung, and Larry Smith. Intel virtualization technology. Computer, 38(5):48–56, 2005.

[3] Irfan Habib. Virtualization with kvm. Linux Journal, 2008(166):8, 2008.

[4] Peter Mell and Tim Grance. The nist definition of cloud computing. National Institute of Standards and Technology, 53(6):50, 2009.

[5] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. A view of cloud computing. Communications of the ACM, 53(4):50–58, 2010.

[6] Ajay Gulati, Ganesha Shanmuganathan, Anne Holler, and Irfan Ahmad. Cloud-scale resource management: challenges and techniques. In Proceedings of the 3rd USENIX conference on Hot topics in cloud computing, pages 3–3. USENIX Association, 2011.

[7] Karissa Miller and Mahmoud Pegah. Virtualization: virtually at the desktop. In Proceedings of the 35th annual ACM SIGUCCS fall conference, pages 255– 260. ACM, 2007.

[8] Xiaofei Liao, Hai Jin, Liting Hu, and Haikun Liu. Towards virtualized desktop environment. Concurrency and Computation: Practice and Experience, 22(4):419–440, 2010.

[9] Su K, Wang Z, Lu X, et al. An original-stream based solution for smoothly replaying high-definition videos in desktop virtualization systems[J]. Journal of Visual Languages & Computing, 2014, 25(6): 676-683.

[10] Jiewei Wu, Jiajun Wang, Zhengwei Qi, and Haibing Guan. Sridesk: A streaming based remote interactivity architecture for desktop virtualization system. In Computers and Communications (ISCC), 2013 IEEE Symposium on, pages 000281–000286. IEEE, 2013.

[11] Gupta P, Gupta S. Mobile cloud computing: The future of cloud[J]. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, 2012, 1(3): 134-145.

[12] Ballagas R, Rohs M, Sheridan J G, et al. Byod: Bring your own device[C]//Proceedings of the Workshop on Ubiquitous Display Environments, Ubicomp. 2004, 2004.

[13] Miller K W, Voas J M, Hurlburt G F. BYOD: Security and Privacy Considerations[J]. It Professional, 2012, 14(5): 53-55.

[14] Ghosh A, Gajar P K, Rai S. Bring your own device (BYOD): Security risks and mitigating strategies[J]. Journal of Global Research in Computer Science, 2013, 4(4): 62-70.

[15] Tristan Richardson, Quentin Stafford-Fraser, Kenneth R Wood, and Andy Hopper. Virtual network computing. *Internet Computing, IEEE*, 2(1):33–38, 1998.

[16] Windows remote desktop protocol (RDP), http://msdn.microsoft.com/en-us/library

[17] Lureau, Marc-André (11 March 2016). "ANNOUNCE: spice-protocol release 0.12.11". spice-devel (Mailing list).

[18] Ricardo A Baratto, Leonard N Kim, and Jason Nieh. Thinc: a virtual display architecture for thin-client computing. In *ACM SIGOPS Operating Systems Review*, volume 39, pages 277–290. ACM, 2005.

[19] Hdx of citrix xendesktop, http://hdx.citrix.com/hdx.

[20] R. A. Baratto, S. Potter, G. Su, and J. Nieh, "MobiDesk : Mobile Virtual Desktop Computing Categories and Subject Descriptors," in MobiCom,2004.

[21] "PC-over-IP," http://www.teradici.com/.

[22] https://www.samba.org/