Computing and Informatics, Vol. 24, 2005, 167–181

# MVNC: A MULTIVIEW NETWORK COMPUTER ARCHITECTURE[*]

Hongliang Yu, Weimin Zheng, Meiming Shen

*Department of Computer Science and Technology*
*Tsinghua University*
*100084, Beijing, China*
*e-mail:* hlyu@tsinghua.edu.cn

Manuscript received 18 August 2004; revised 8 March 2005
Communicated by Ivan Plander

**Abstract.** In this paper, MVNC, a multiview network computer system for a high usability thin-client computing environment, is introduced. MVNC uses a revised SBC model to offer a new framework for thin client computing. MVNC can be used as a full functional Windows machine, or used as a Linux workstation, or a graphic terminal. Its multiview work style is achieved by the attempts on GUI seamless integration technology, device integration technology and local video playback support. MVNC is implemented in an embedded Linux environment using a MIPS-4KC microprocessor. Test results on video application show that MVNC system uses its client hardware more efficiently and the load of MVNC server is lightened.

**Keywords:** Network computer, thin client, server based computing

## 1 INTRODUCTION

Thin computing is becoming a key research area for future computing paradigms. With the development of computer technology, the speed of computer network is improving dramatically, network based applications have been developing very fast too. Thin-client computing is originated in this background. "Thin client" concept, where most processing will be done by server and the client mostly takes charge of

[*] Supported by the National Natural Science Foundation under Grant No. 60433040

displaying screen image and sending input events to server, has been changing all the ways too. According to the Moore's law, thin clients will become increasingly powerful than before, while considerable research has been done on providing a base-level of functionality in thin system comparable to stand-alone PCs or workstations. So, we should find some new ways in which the power of thin systems can be exerted.

In this study, we make some attempts for providing a more convenient and suitable way for using thin systems. We introduce our multiview network computer system(MVNC). MVNC offers a new framework for network computing, it is based on one of the most popular thin-client computing architecture, namely server based computing. We say "multiview" because with our seamless Graphic-User-Interface seamless integration technology, we can use MVNC in different modes: just like a Windows PC, like a Linux workstation, or as a graphic terminal like traditional thin clients, while getting improved price/performance ratio, better maintainability and upgraded ability.

The rest of the paper is organized as follows: Firstly, we will analyze the problems existing in traditional thin client system. After that, we look back to the development of computing and find how thin-client computing originated and developed from it. Then, we design a new model for MVNC; this model is more functional and easy-to-use. Finally, the implementation details of MVNC are introduced with some test results.

## 2 THE DESIGN OF MVNC

### 2.1 The Problem

Although existing thin client systems offer solid baseline features, support for advanced features such as video, local storage devices is quite poor [1, 2]. On the other hand, people have been used to work in Windows based systems, and thus it is important for thin client systems to have Windows like graphic user interface. Furthermore, as the computing power of thin client systems is developing so fast, it is very reasonable to use this computing power more sufficiently but not using thin clients as graphic terminals only.

In today's market, the most successful thin client technology is the server based computing (SBC) technology. It is becoming mature now [3], SBC allows the deployment of traditional client-server applications over the Internet and WANs while reducing maintenance and support costs, and it can provide a Windows like graphic user interface, if required. The drawbacks of SBC will be discussed below.

Firstly, in most SBC system, no differentiation is made between video data and regular graphics data. This means when server decodes the video data into separate frames and displays it, these frames will be encoded as some static pictures by SBC scheme and transferred to client machines. This consumes lots of computing and network resources on SBC server. Some systems, like Citrix VideoFrame, recognize video data as a different type and supports it through a proprietary codec format.

The stream is decoded from its original format, encoded for transmission using the proprietary codec and decoded again at the client side. In this process, the server will be burdened with some unnecessarily transcoding operations [1].

Secondly, remote display protocol has become essential to server based computing. There are many such protocols like RDP, ICA, AIP, VNC, for etc., but only windows based ICA and RDP (developed from ICA) are actually working well for the support of Microsoft. Other protocols like VNC perform well on UNIX, but quite poorly in Windows environment. While some work has been done to solve the video playback in remote display system [4, 5, 6] like VNC, it is quite meaningful to find some common ways for all main systems (especially RDP on UNIX) to share the advantages of SBC.

Finally, in thin systems using SBC technology, lots of thin clients communicate with the thin server. The only task of these thin clients is displaying the vision which is produced by the server, and all real computing works are finished in the server. This fully centralized computing model will result in the overload of thin server while the power of thin clients are squandered.

## 2.2 Different Computing Paradigms

We propose an alternate solution for the problems described above. For better understanding of our ideas, we firstly look back to the development of computers. In fact, the evolution of computing has great impact on the way that server based computing does. If you want to change it, you must first know how it originated.

Computing technology has evolved considerably over time. Although the process of evolution has been continuous, it can be classified as three distinct computing paradigms [7, 8].

The first computing paradigm, the mainframe paradigm, was popular through the 1960s and 70s (Figure 1). In this paradigm, computer processing power was provided by mainframe computers located in air-conditioned computer rooms. All resources were centralized, the main way to access the mainframe was through some character-based terminals; and the speed of the links between terminals to the mainframe was quite slow. This paradigm had suffered greatly from the costs of maintaining and administrating the system, and from the poor price-to-performance ratio too.

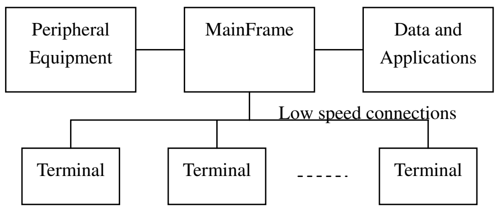


Fig. 1. Host centric computing

The second computing paradigm is involved with powerful PCs and workstations (Figure 2). These standalone computers were single-user systems that executed a wide range of applications; they have their own computation and storage units, so they can work independently. With the advent of fast networking technologies, PC-LANs became more popular, which connect the PCs and workstations together. Client/server computing originated from this; a server in the LAN can provide centralized services to other PCs or workstations. Generally, client machines store and run applications locally while the server provides some file and print services. The pioneers of thin-client computing, such as InfoPad [9] system and Java NC system [11], are some kinds of client/server computing indeed.

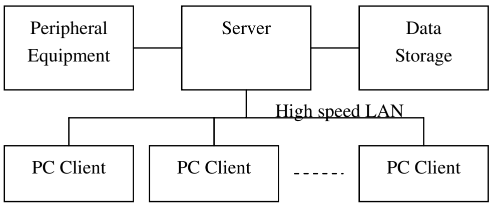


Fig. 2. Desktop centric computing

Network-centric computing developed in early 1990s (Figure 3). Users' machines access both applications and data on networked servers in this paradigm. The most influential form in network-centric computing is Internet computing, which changed our application-development and content-delivery modes. The Internet has shifted the distributed computing paradigm from the traditional closely coupled form to a loosely coupled one [8]. With the development of Internet computing, more and more applications are moved from client/server mode to web-based browser/server mode, which needs no client software installations in client machines. This is another kind of thin-client system.

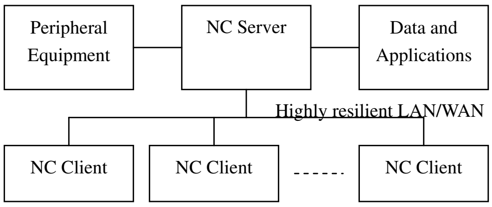


Fig. 3. Network centric computing

However, commercial requirements and realities have revealed that professional applications are not always compatible with a web-based mode. For example, certain applications required the transmission of huge amounts of data between server and client, and cannot be redesigned for web-based mode. For this and other reasons, many businesses chose not to redevelop their applications and continue to use the traditional client-server model. Then, a new thin-client computing mode—server-based computing (SBC) [10]—was populated to fill in the gap.

In SBC, a terminal server, one that is remote from the client terminal, deploys, manages, supports, and wholly runs the client part of the application. This technology uses a multi-user operating system as a terminal server running numerous client sessions and a protocol that lets the thin client terminal display the user interface. The only application resident on the client terminal is the one displaying or reproducing the user interface. From the user's point of view, this technology combines the advantages of a thin client with the spread of functions offered by the client-server solution. A SBC system supports many client platforms, uses low bandwidth, and works with any type of client server application.

## 3 THIN COMPUTING MODEL

Thin client system is developing with the tide of network computing. One of the oldest thin client system is network computer (NC) [11] promoted by Sun and Oracle. It is designed to a platform running with Java. This NC system downloads Java applications through network, and executes them locally; it is a kind of client/server computing. There is no storage device in Java NC, all data and applications are stored in corresponding NC server. Java based NC is relatively cheap, and easy to manage. However, it is a totally different system from the Windows system used by most people; this is the main reason why Java NC failed.

Browser/server based computing is another kind of thin computing. It is developed from client/server computing which is popular in network-centric computing times. The main advantage of Browser/server model is that no software is needed to be installed beforehand. Its easy-to-use GUI ensures wide acceptance.

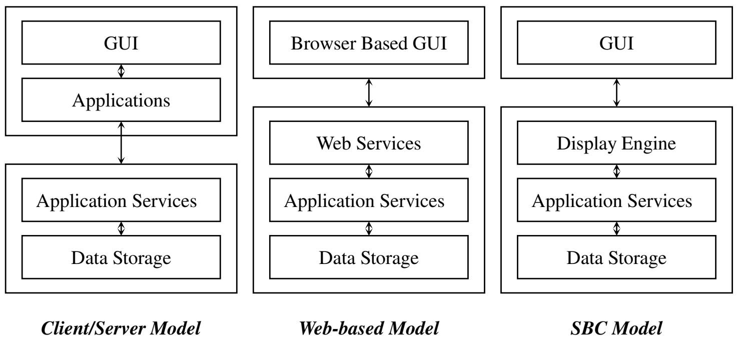


Fig. 4. Three different thin-client computing models

We have known that client-server computing based thin-client system like Java NC has no software installed locally, but they can download the operating system and application software from the server. After downloading, they can execute such applications locally. For example, if you have suitable software written in Java, you can do word processing, draw pictures, or even play videos locally. But such kind of thin client systems are generally considered as an unsuccessful initiative, because they cannot be compatible with Windows system, which is widely used and accepted. SBC system generally uses the Windows system, so they become more successful. However, for the lack of client local processing capability, the server is often overloaded. So, if we can combine the advantages of two computing models, we will have a good answer.

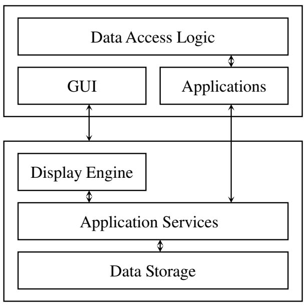


Fig. 5. Revised SBC model

We can see from Figure 5 that this computing model is developed from server based computing. Thin clients connect to a server and display the graphic-user-interface of the server. But they have two functions indeed. One is to act as a graphics terminal; the other is running as a full functional machine. Thin client functions are extended through local application execution capabilities. In this mode, thin client can even have a data access logic, which can store private data through network.

The model is not too complex; we will show it in detail below with our implementation of a prototype MVNC system.

## 4 IMPLEMENTATION OF MVNC

Thin client of server-based computing runs UNIX or Windows operating system; as for Windows, Microsoft has all the solutions for it. We mainly solve the problem for clients running UNIX. Then we assume that Windows system is running on the server, so the SBC client can display a windows-like interface. Furthermore, since Linux has become more than a hobbyist operating system for interested persons to fuzz around with, our implementation also considers using a Linux based NC server.

We will describe the current prototype implementation of the system below.

### 4.1 Hardware

We designed an embedded system for server-based computing, which is named MVNC. It is powered by our embedded micro-processor, namely THUMP embedded microprocessor running at 400 MHz. THUMP embedded microprocessor uses a MIPS instruction set, and is fully compatible with MIPS-4KC.

IT8172G is used in our mainboard. It performs high speed, high performance core logic control and data accessing between the CPU bus, system SDRAM bus, and PCI bus.

### 4.2 OS and Applications Installation

As we have a thin client system, there is no hard disk in our client machine. So, we have designed two booting modes for it: one is booting from a FLASH module, the other is booting from network.

An optional installed FLASH memory (8 MB) can be used to store the OS kernel and some critical applications, which can be used as the booting source. The advantage of this mode is the speed, but the total cost of thin client hardware will be added too.

We have a BIOS like module named PMON in our hardware; it can be used to boot the system from network. In this mode, thin client will get its IP address through a DHCP request; then, TFTP protocol is used to download the OS kernel from the server. Finally, the system is booted and some remote disks will be used locally through NFS.

### 4.3 Graphic-User-Interface Seamless Integration

We designed a framework named graphic-user-interface seamless integration to combine the advantages of traditional server-based computing and classical client-server computing. It is essential to get multiview in MVNC.

MVNC has three working modes, one is used as a Linux workstation, the other is used as a thoroughly Windows based PC, and the final one is used as a graphic terminal. This just reflects the meaning of multiview. The implementation of a graphic terminal is quite simple and traditional, we will not mention it any more in this paper.

If the user of MVNC wants to use it as a Linux machine, X Window in Linux will be used. In this mode, a X server program will be executed in MVNC client, interacting with X client programs (such as Open Office, Mozilla, etc.) running remotely in MVNC server.

It is relatively difficult to use MVNC as a Windows PC. Our prototype is built on top of rdesktop [12] for UNIX. Rdesktop is an open source UNIX X11 client for Windows NT Terminal Server, capable of natively speaking its Remote Desktop Protocol (RDP) in order to present the user's NT desktop. Unlike Citrix ICA, no server extensions are required.

After being booted, our thin client machine will be connected to a Windows server automatically through rdesktop. So, we can log on to the server immediately. After that, a windows graphic-user-interface will be shown on thin client display, but it is partly changed by us.

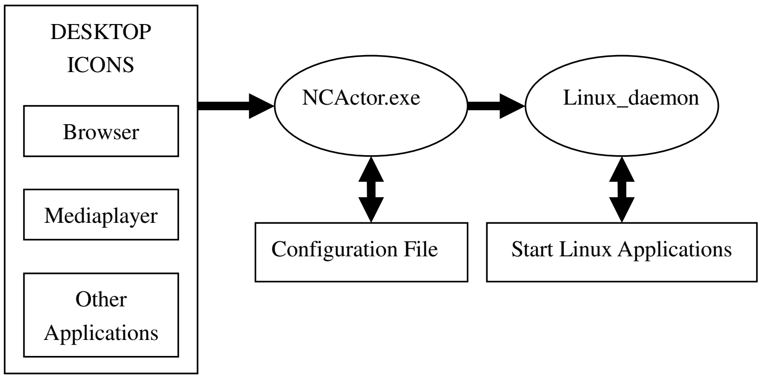


Fig. 6. Graphic-User-Interface seamless integration

Firstly, we have modified some of the system icons and shortcuts. In traditional server based computing, all icons and shortcuts on thin client displays mean some corresponding server applications. In our system, they will be divided into two categories; one for server applications, the other for some local applications on thin client machines. We design a client-server style mechanism for the execution of local applications in our thin client machine. Any request for the execution of corresponding applications of these icons and shortcuts will be passed to a process named PIQUET (NCActor.exe) in SBC server; then, this process will send the request to a DAEMON (Linux_daemon) running on thin client through network; after receiving the request, the DAEMON will start the correct application on thin client machine, and commit the request at the same time. After that, any window operations like window maximize, window minimize, window close and window move will all be communicated between the PIQUET and the DAEMON, in order to keep the whole graphic-user-interface seamless.

Secondly, some file associate relations are modified too. For example, if you find a video file in Windows explorer, you may double-click it for a playback. Traditional server based computing use a windows media player for this task, but in our system, another media player running on thin client will be in charge. Other applications like Mozilla will be used in place of IE too. As for users, the real execution locations of applications are not so important; the important thing is: they can use them in one style.

### 4.4 Local Video Processing

We use VideoLAN [13] for video streaming and playback in our system. VideoLAN is free software, and is released under the GNU General Public License. The VideoLAN project targets multimedia streaming of MPEG-1, MPEG-2, MPEG-4 and DivX files, DVDs, digital satellite channels, digital terrestial television channels and live videos on a high-bandwidth IPv4 or IPv6 network in unicast or multicast under many operating systems. VideoLAN also features a cross-plaform multimedia player, VLC, which can be used to read the stream from the network or display video read locally on the computer under all main operating systems.

We design a streaming server for better using of VideoLAN in our system. In our revised SBC model, local applications in thin clients communicate with thin server in client-server mode. We can consider the local applications as a streaming media client, and thin server as a streaming server. The easiest way to implement a streaming server is through the HTTP service of Windows system, but it is not so reliable. If the server could not send enough media data to the client, video playback will be paused. Then we studied from traditional video on demand technologies. Interval caching [14] is used in our system. The final stage of our implementation is a file mapping scheme; we consider the whole file system in thin server as a video database, any request to the video file will be redirected through the PIQUET and the DAEMON described earlier. This is another main step to approach our seamless GUI.

### 4.5 Local Devices Integration

#### 4.5.1 Theories

There is no hard disk or other local devices in traditional network computer such as Java computer or SBC client. But, if you want to use network computer more conveniently, local devices must be supported. SMB protocol and GUI seamless integration technology are used for our local devices integration. With it, we can use USB disks or print files locally and freely in our network computers, just like using a Windows based PC too.

Server Message Block (SMB) protocol is a protocol for sharing files, printers, serial ports, etc. between computers. The SMB protocol can be used over the Internet on top of the TCP/IP protocol or other network protocols such as Internetwork Packet Exchange (Novell IPX) and NetBEUI. Microsoft Windows operating systems since Windows 95 include client and server SMB protocol support. Microsoft has offered an open source version of SMB for the Internet, called the Common Internet File System (CIFS), which provides more flexibility than existing Internet applications such as the File Transfer Protocol (FTP). For Unix systems, a shareware program, Samba [15], is available. Samba uses the TCP/IP protocol that is installed on the host server. It allows the host to interact with a Microsoft Windows client or server as if it were a Windows file and print server.

We install the Samba server software in every NC client system, and all local resources such as USB storage devices, hard disks, CD-ROMs or printers are all added to Samba, which provides file sharing and printer sharing services using SMB protocol. NC server is configured to act as a SMB client. It sends SMB request to every NC client, and interacts with them afterwards. This way, when we logged into the Windows based NC server, we can use or control the storage devices or printers resided in NC clients. For convenience, we can map the sharing folders into a network disk driver, or the network printer into a local printer too. Based on these, in MVNC, NC client logged into the remote Windows server, and used SMB protocol to access back to their own devices. From the point of view of a common NC user, what he/she feels is only a Windows based computer, and all local devices can be used easily and freely.

### 4.5.2 Work flow

Our implementation on local devices integration includes two main parts: lpdaemon in Linux system and corresponding applications in Windows system (such as USB disk application and printer application).

The lpdaemon is designed to start automatically when NC client is powered on. It listens to the specified port and waits for the requests from NC server.

In NC server, when a special application is activated, the function WTSQuery-SessionInformation will be called to get the IP address of corresponding NC client. Then, a connection is built between the NC server and lpdaemon running in NC client. The lpdaemon will be notified about the device type requested, and check for the availability of the device locally in NC client. In Linux system, files in the `/dev/` path indicate all existing devices, for example, file prefixed by `hd` indicates IDE hard disk, `sd` indicates USB storage device, and `lp` for printers.

After that, different devices will have different work flows. For USB disks, usbdev application in NC server will load the remote device via function call WNet-AddConnection2, and a tray program will be activated to indicate a USB storage device icon at the right side of the task bar in Windows GUI. The tray program has only one function: to safely store device removal. When users select the tray program on Windows desktop, the tray program will communicate with the lpdaemon, and interact with it to finish the storage device removal action. After that, an operation successful message will be shown to the NC user by the tray program.

We follow a similar work flow when dealing with the printer devices. The common rules are: finishing the basic functions by communicating with lpdaemon; simulate corresponding GUI actions in Windows desktop.

## 5 TEST RESULTS

Albert Lai [2] used a novel, non-invasive slow-motion benchmarking technique to evaluate the performance of several popular thin-client computing platforms. S. Jae Yang [1] gave sufficient results to value the effectiveness of a number of server based computing design and implementation choices across a broad range of thin-client platforms and network environments too.

| Usage | Hardware | Software |
|---|---|---|
| NC Client | THUMP-CPU 400 MHz, 64 M RAM | Linux 2.4.18 |
| NC Server | Pentium 4 2.8 GHz, 1 G RAM | Windows 2003 Server |

Table 1. MVNC Testbed

Our implementations are mainly complementary to RDP based server based computing. So, we only care about the differences. We use a isolated network testbed to measure the performance of MVNC for video playback, which always have unsatisfactory performance result in thin client systems. Our test environment is configured as shown in Table 1.

| Video Source | Local Playback | Windows RDP Client | MVNC |
|---|---|---|---|
| 38 Kbps DIVX4 | 4.9 % | 11.2 % | 1.6 % |
| 50 Kbps DIVX4 | 5.7 % | 11.6 % | 1.8 % |
| 56 Kbps DIVX3 | 5.7 % | 12.6 % | 1.7 % |
| 73 Kbps DIVX5 | 5.9 % | 12.1 % | 1.8 % |
| 166 Kbps MPEG1 | 3.2 % | 19.6 % | 2.1 % |
| 193 Kbps MPEG4 | 5.3 % | 15.3 % | 2.1 % |
| 334 Kbps MPEG1 | 3.6 % | 16.7 % | 4.2 % |
| 610 Kbps MPEG2 | 14.1 % | 40.5 % | 5.6 % |

Table 2. Video playback CPU utilization

| Video Source | Local Playback | MVNC |
|---|---|---|
| 38 Kbps DIVX4 | 6.6 M | 0.1 M |
| 50 Kbps DIVX4 | 6.9 M | 0.1 M |
| 56 Kbps DIVX3 | 6.8 M | 0.1 M |
| 56 Kbps DIVX5 | 7.1 M | 0.1 M |
| 73 Kbps DIVX5 | 7.1 M | 0.1 M |
| 166 Kbps MPEG1 | 6.3 M | 0.1 M |
| 193 Kbps MPEG4 | 13.1 M | 0.1 M |
| 334 Kbps MPEG1 | 6.3 M | 0.1 M |
| 610 Kbps MPEG2 | 11.3 M | 0.1 M |

Table 3. Video playback memory usage

Our tests were finished at $1024 \times 768$ 16 bit color display resolution. We test the decoding process in three modes. Firstly, we decode the file on NC server locally; secondly, we connect to NC server through RDP and run the decoding; finally, we connect to NC server from our MVNC and do the same. Our main results are shown in Tables 2, 3. We can see that the CPU utilization and memory usage of NC server are all greatly reduced, because neither decoding nor rendering are needed on the NC server in the latter mode. With much lower resource consumption in NC server, we can support more NC clients with a single NC server; this is very important to thin client system where the price/performance ratio is critical.

## 6 RELATED WORKS

Thin client system has been evolving over years. As one of the most important thin computing paradigms, the first version of server-based computing was provided by the X Window system, it was originally developed for UNIX and enabled interactive applications running on large servers to accessed from low-cost workstations.

However, the Windows system dominated in the world soon. The server-based computing architecture from CITRIX [10] allows a variety of remote computers, regardless of their platform, to connect to a Windows NT terminal server to remotely access a powerful desktop and its applications. A server called MetaFrame runs under Windows NT in the desktop machine and communicates with the thin clients executing at the remote computers using the Independent Computing Architecture protocol (ICA). The ICA client and the MetaFrame server collaborate to display the virtual desktop on the remote computer screen. They also collaborate to process mouse and keyboard events, and to execute programs and view data stored at the server. All executions are remote and none takes place at the client's portable computer.

A research project at Motorola [16] extended CITRIX's thin client architecture so that it is optimized in the wireless environment. The work pointed out that bandwidth limitation is not as detrimental to the thin client performance as network latency. This is because the thin clients' use of bandwidth is limited. Other server-based computing implementation includes Remote Display Protocol (RDP) of Microsoft [17], Tarantella AIP of Tarantella [18], and Virtual Network Computing (VNC) of AT & T [19, 20]. All these systems are quite similar, that is to say, all processing will be done on the server, and terminal devices work as a remote display. Each system has each proprietary protocol and data compression between server and terminal to reduce both network traffic and terminal side processing. Reduction of network traffic is usually achieved by sending updated image only.

The Infopad [9] project gives a good attempt on the topic of remote access to multimedia content. Other systems can even provide remote access to 3D content, including SGI's VizServer [21], WireGL and Chromium [22]. Java is back to thin client area too, J2EE and ASP.NET technology [23] are all included. The most recent work in thin client system includes THINC [6], which transparently maps high-level application display calls to a few simple low-level commands which can be implemented easily and efficiently.

## 7 CONCLUSIONS

In traditional server based computing, a terminal server deploys, manages, supports and wholly runs the client part of application. Thin client only takes charge of the display of the graphics user interface of the running result. With the development of embedded systems, the power of thin client devices is greatly boosted up.

We have introduced MVNC, a multiview network computer system for a high usability thin-client computing environment. MVNC uses a simple, revised SBC model to offer a new framework for server based thin computing. MVNC can be used as a full functional Windows machine, and this ensures good usability. It can be also used as a Linux workstation or a graphic terminal. This multiview work style is achieved by our efforts on GUI seamless integration technology, device integration technology and local video playback support. With MVNC, the thin client hardware is used more efficiently, too.

Since MVNC's characteristics of easy using is hard to measure, we only measured MVNC's performance in video application. Our experimental results in video application show that the load of MVNC server is greatly lightened. This also improves the price/performance ratio of the whole system.

## REFERENCES

[1] Jae Yang, S.—Nieh, J.—Selsky, M.—Tiwari, N.: The Performance of Remote Display Mechanisms for Thin-Client Computing. Proceedings of the 2002 USENIX Annual Technical Conference, Monterey, CA, June 10–15, 2002, pp. 131–146.

[2] Lai, A.—Nieh, J.: Limits of Wide-Area Thin-Client Computing. Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS 2002), Marina del Rey, CA, June 15–19, 2002, pp. 228–239.

[3] Volchkov, A.: Server-Based Computing Opportunities. IT Professional, Vol. 4, 2002, No. 2, pp. 18–23.

[4] Christiansen, B. O.—Schauser, K. E.—Munke, M.: Streaming Thin Client Compression. Proceedings of Data Compression Conference 2001, pp. 223–232.

[5] Kuo, Ch.-Ch.—Ting, P.: Design and Implementation of a Network Application Architecture for Thin Clients. Proceeding of the 26th Annual International Computer Software and Applications Conference (COMPSAC '2002).

[6] Baratto, R. A.—Nieh, J.—Kim, L.: THINC: A Remote Display Architecture for Thin-Client Computing. Technical Report CUCS-027-04, Department of Computer Science, Columbia University, July 2004.

[7] Revett, M.—Boyd, I.—Stephens, C.: Network Computing: A Tutorial Review. Electronics & Communication Engineering Journal, Vol. 13, 2001, No. 1, pp. 5–15.

[8] Ahmad, I.: Network Computers: The Changing Face of Computing. IEEE Concurrency, October–December, 2000.

[9] Seshan, S.—Le, M. T.—Burghardt, F.—Rabaey, J.: Software Architecture of the Infopad System. In Proceeding of Mobidata Workshop on Mobile and Wireless Information Systems, New Brunswick, NJ, November 1994.

[10] `http://www.citrix.com`, 1998.

[11] Java NC Computing, `http://java.sun.com/features/1997/july/nc.html`.

[12] `http://sourceforge.net/projects/rdesktop/`.

[13] http://www.videolan.org.

[14] Dan, A.—Dias, D. M.—Mukherjee, R. et al.: Buffering and Caching in Large-Scale Video Servers. Proceedings of COMPCON. San Francisco, USA, IEEE Computer Society Press, 1995. pp. 217–224.

[15] http://samba.anu.edu.au/samba/.

[16] Duran, J.—Laubach, A.: Virtual Personal Computers and the Portable Network. In Proceedings of the IEEE Conference on Performance, Communication, and Computing 1999 (Phoenix, AZ). IEEE Computer Society Press, Los Alamitos, CA.

[17] Microsoft RDP & Citrix ICA Feature Overview, http://www.microsoft.com/windows2000.

[18] http://www.tarantella.com.

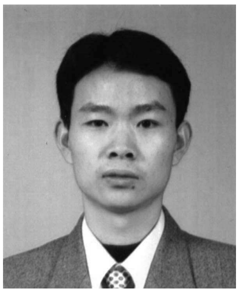
[19] Richardson, T.—Stafford-Fraser, Q.—Wood, K. R.—Hopper, A.: Virtual Network Computing. Internet Computing, IEEE, Vol. 2, 1998, No. 1, pp. 33–38.

[20] Kaplinsky, K. V.: VNC Tight Encoder-Data Compression for VNC. Modern Techniques and Technology, 2001. MTT 2001. Proceedings of the 7th International Scientific and Practical Conference of Students, Post-graduates and Young Scientists, 26 Feb.–2 March 2001.

[21] SGI OpenGL Vizserver. http://www.sgi.com/ software/vizserver/.

[22] Humphreys, G.—Houston, M.—Ng, R.—Frank, R.—Ahern, S.—Kirchner, P.—Klosowski, J. T.: Chromium: A Stream Processing Framework for Interactive Rendering on Clusters. In SIGGRAPH 2002.

[23] Grundy, J.—Wei, Z. et al.: An Environment for Automated Performance Evaluation of J2EE and ASP.NET Thin-Client Architectures. Proceedings of Software Engineering Conference 2004, Australia, pp. 300–308.



**Hongliang Yu** works in the Institue of High Performance Computing, Department of Computer Science and Technology, Tsinghua University, Beijing, China. He received the M. Sc. degree in computer science from Tsinghua University in 1998. He is currently working toward the Ph.D. degree in the Department of Computer Science and Technology at the Tsinghua University. His research interests include embedded system, active storage, and streaming of media.

**Weimin ZHENG** is a professor in the Department of computer science and technology, Tsinghua University, Beijing, China. His major research interests include parallel/distributed and cluster computing, and run-time system design for parallel processing systems. Zheng and his group faculty, research staff members are currently working on a number of R & D projects supported by the Natural Science Foundation of China and the National High-Tech Program and Zheng is also the Managing Director of the Chinese Computer Society.

**Meiming SHEN** is professor in the Department of Computer Science and Technology, Tsinghua University, Beijing, China. Her research interests include parallel and distributed systems, p2p system and streaming of media.