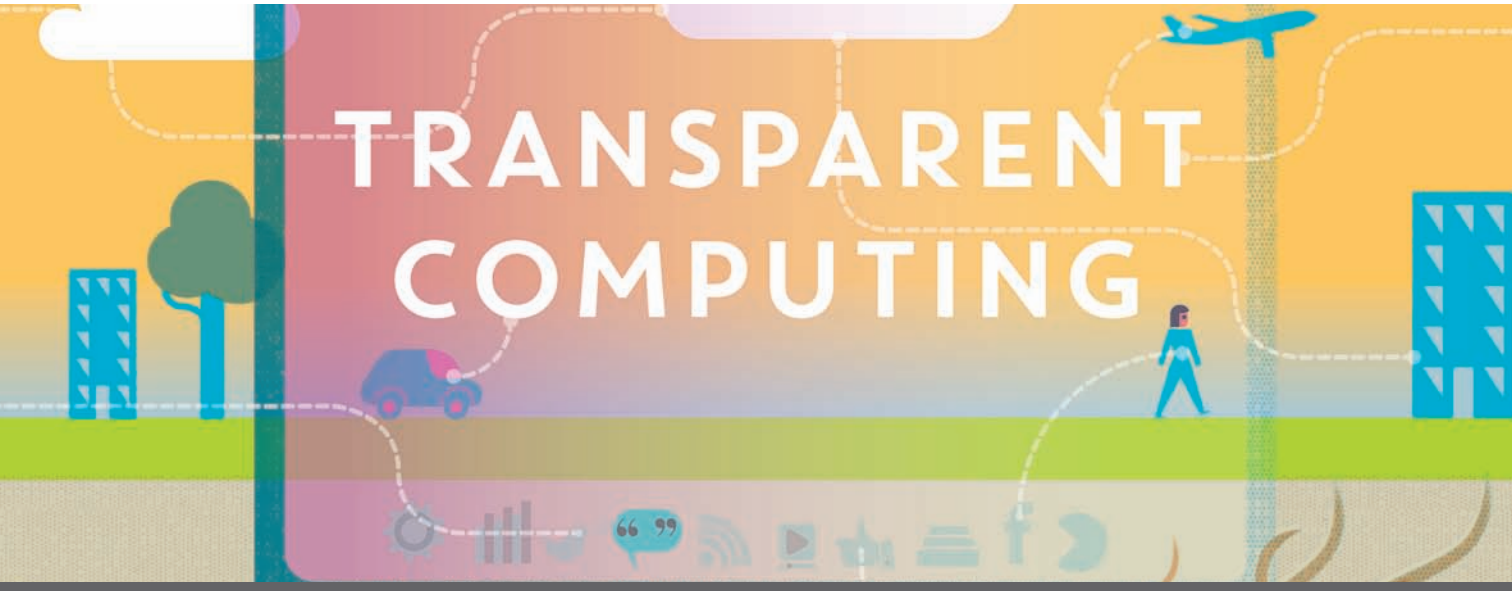

Transparent Computing:
A Promising Network Computing Paradigm

Yaoxue Zhang, Kehua Guo, and Ju Ren | Central South University, China
Yuezhi Zhou | Tsinghua University, China
Jianxin Wang and Jianer Chen | Central South University, China

By adopting the transparent computing paradigm, user terminals are becoming more lightweight with enhanced security, improved energy efficiency, and cross-platform capability. A comprehensive survey indicates future directions of transparent computing, from traditional terminals to mobile devices. New challenges and potential research directions for applying transparent computing to mobile devices could foster further study.

In the past 10 years, the emergence of new technologies such as cloud computing and big data have changed the core functions of computers and the Internet from computing and communicating to collecting, storing, analyzing, and using various data and services. Simultaneously, we have witnessed the proliferation of mobile devices and are heading toward the Internet-of-Things (IoT) era, where lightweight and mobile devices are the dominant terminals on the Internet and in our daily life. In 2014, the global revenue of the mobile Internet totaled US$3.3 trillion, and smartphone shipments exceeded 1.3 billion units (approximately three times the volume of PC shipments; www.catr.cn/kxyj/qwfb/bps/201512/P020151211378960738048.pdf). Clearly, the network computing environment is gradually evolving into a mobile Internet in which mobile devices request different services via various wireless communication networks, such as Wi-Fi, cellular, and ad hoc networks.

The dramatic increase in mobile devices and services provides significant opportunities and motivates the evolution of the computing paradigm from PCs to mobile devices. However, new terminals and network environments also introduce new challenges. Lightweight terminals and smart applications (apps)

 1521-9615/17/$33.00 © 2017 IEEE Copublished by the IEEE CS and the AIP

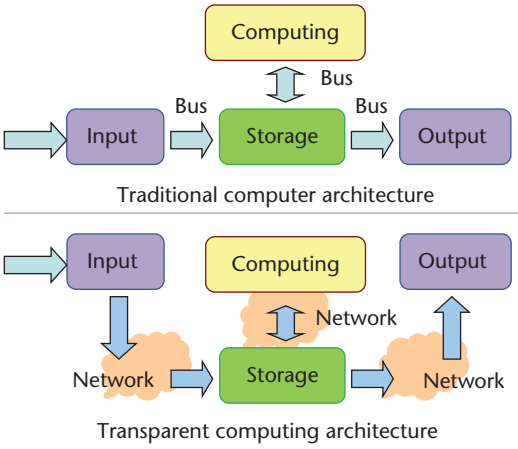

Figure 1. A new paradigm. Transparent computing extends the bus transmission found in traditional computer architectures.

possess such characteristics as mobility, portability, user dependency, immediacy, and privacy. In the mobile Internet era, server-centric computing paradigms (virtual desktops, cloud computing, and big data processing) have been the representative technologies, but they offer only partial solutions to certain problems.[1] On these platforms, multiple virtual operating systems (OSs) run on cloud servers, and users access various resources via a remote desktop from different types of terminals. Computing is performed by servers, whereas terminals exclusively receive and display the processed data.

Although server-centric computing paradigms such as cloud computing have the advantages of easy maintenance, centralized management, and high server utilization, they present limitations and numerous challenges to terminals. First, terminals require a lot of storage and a high processing capability to store and execute OSs and software programs as they communicate with central servers. Second, because of the loss of credibility and controllability in terminals, users might be vulnerable to many types of security attacks, such as data theft, data tampering, and privacy leakage Third, because terminals only display computing or processing results, an inferior user experience could happen when terminals support apps that require frequent user interactions. In short, server-centric computing paradigms only solve problems from the server and network perspective, not from that of users and services.

Transparent computing,[2] which was first proposed in 2004, is a promising solution for mitigating or addressing these challenges. The core idea is that all data and software, including OSs, apps, and user information, are stored on servers, with data computing performed on terminals. Users can obtain heterogeneous OSs and apps from servers and run them at their own terminals without needing to know underlying hardware and implementation details.[3] User interruptions from terminals are redirected to a server through a network connection to request the corresponding instructions and data, which are subsequently executed in a page-streaming pattern. Thus, transparent computing possesses the following four advantages:

- *It reduces terminals' complexity and cost.* Terminals can become extremely simple, nearly bare machines without software, thus making them more lightweight and reducing hardware costs.
- *It improves the user experience.* The terminal is not only responsible for cross-platform display, but it can also run heterogeneous programs from various platforms, thus making it capable of supporting apps that require frequent interactions, such as video games.
- *It offers a high level of security.* We can track program execution routes from servers to terminals, and because instructions on terminals are dumped after use, system security is enhanced.
- *It offers cross-platform capability.* With a transparent computing platform, it's possible to develop apps that work across hardware and software platforms. For example, iOS apps can be compatible with Android apps, which lets users freely access services hosted on networks that cross heterogeneous software and hardware platforms.[4]

Server-centric computing paradigms such as cloud computing are based on parallel virtualization and solve the issue of data in the cloud, whereas transparent computing emphasizes data storage on servers and computation on terminals, streaming both execution and heterogeneous services support for heterogeneous terminals. However, server-centric computing paradigms and transparent computing aren't mutually exclusive; cloud resource management and big data processing can also be applied on the server side of transparent computing. In essence, transparent computing extends the bus transmission found in traditional computer architecture to the network (see Figure 1).

Given the inherent advantages of transparent computing, a large number of researcher and companies have focused on the topic over the past decade. Many transparent computing standards and products

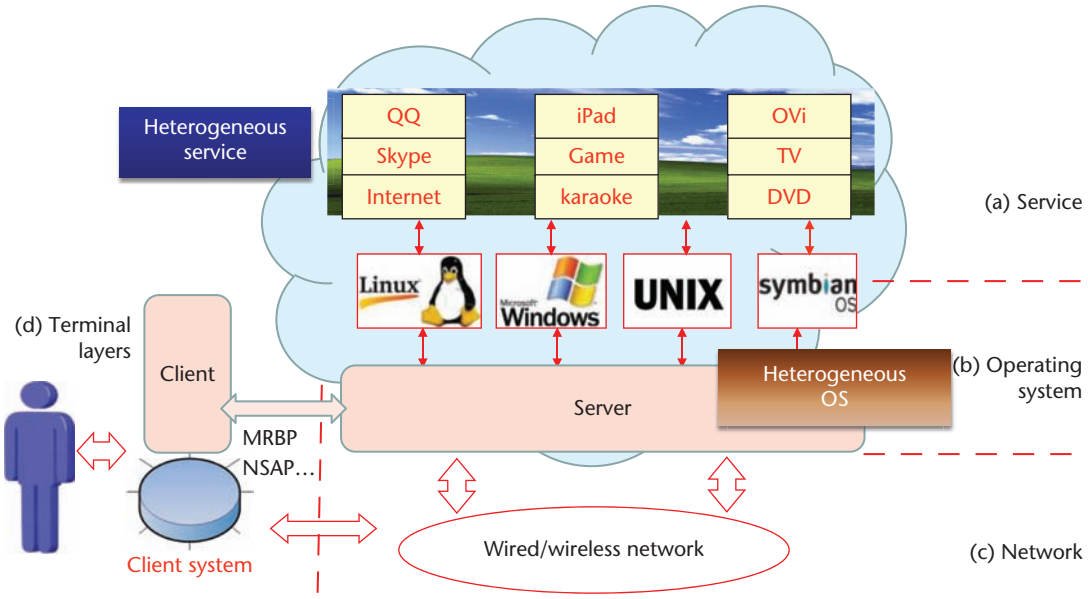

Figure 2. Transparent computing. The four-layer architecture consists of (a) service, (b) operating system, (c) network, and (d) terminal layers.

have emerged, attracting significant attention from both industry and academia. In 2007, a research team at Intel designed a transparent computing platform called UEFI (Unified Extensible Firmware Interface) and developed several apps on it by utilizing Intel's new-generation BIOS (Basic Input Output System).[5] Academic researchers have also developed transparent computing systems to support heterogeneous hardware architectures based on UEFI.[6] During the 2012 Intel Developer Forum, Renee James, executive vice president and general manager of the company's Software & Services Group, proclaimed, "The next era of computing is transparent computing" (http://download.intel.com/newsroom/kits/idf/2012_fall/pdfs/IDF2012_Renee_James.pdf).

Transparent computing has garnered attention from the media as well. In 2012, ScienceDaily posted, "An Operating System in the Cloud: TransOS Could Displace Conventional Desktop Operating Systems" (https://www.sciencedaily.com/releases/2012/10/121009111944.htm), and TechEye posted, "Researchers to Push OS into the Cloud" (www.techeye.net/software/researchers-to-push-os-into-the-cloud). TechNewsDaily, ScienceNewsline, iSGTW, gizmag, Xataka, and eWeek have also reported on the topic.

In this article, we provide a comprehensive survey of transparent computing and indicate some future directions for it, from traditional terminals to mobile devices.

Transparent Computing: Concept, Architecture, and Key Technologies

Transparent computing lets users enjoy services via on-demand network access with any type of device, without needing to know the location of OSs, middleware, and apps.[3] This paradigm separates data storage and program execution in different computers connected by communication networks. Data and software (including OSs) are stored and scheduled by servers but executed on users' terminals.[7] No OS, middleware, or application programs are installed on users' terminals; rather, they're dynamically loaded from the server through the network when users submit requests.[2]

As Figure 2 shows, transparent computing has an architecture consisting of service, OS, network, and terminal layers.

In the service layer, software and user data are stored on servers. Services can be heterogeneous and are automatically associated with different underlying Oss; users can request any service regardless of the underlying OS.[2] Services can be deployed by download or user install, and they generally include a wide variety of options, such as games, instant messaging tools, and smart home apps.

The OS layer stores heterogeneous OSs. To support heterogeneous services, underlying OSs must be maintained by the servers. When a user requests a service, the servers automatically choose the underlying OS and select the OS kernel, thereafter

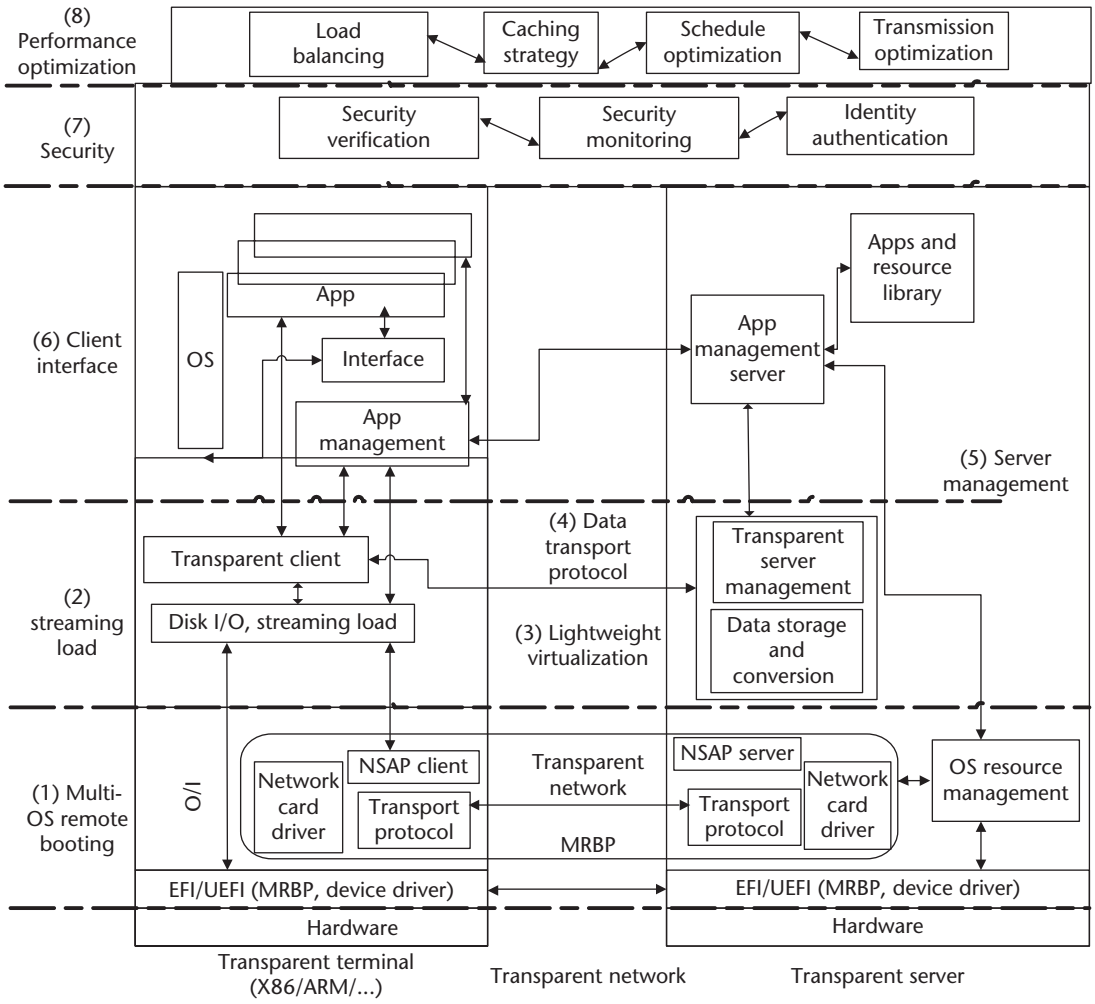

Figure 3. Key technologies in transparent computing. There are eight technical points in a transparent computing platform, namely, multi-OS remote booting, streaming loading, lightweight virtualization, data transport protocols, server management, client interface, security, and performance optimization.

transmitting the kernel together with any requested service apps to the user's terminal. Generally, both services and OSs are stored on the servers that are responsible for managing OSs, software, and user data. In transparent computing, computation and storage are spatiotemporally separated. Suitable OSs and software are dynamically dispatched from the servers to users' terminals and executed on the terminals in a buffer-enabled block or a streaming way.

The network layer controls communications and data transmission between servers and users. Software, OSs, and data are formed as blocks or streams that flow from the servers to users' terminals. Some communication protocols such as MRBP (Multi-OS Remote Booting Protocol)[8] and NSAP (Network Storage Access Protocol)[9] have been developed to support remote booting and block transmission.

The fourth layer is responsible for receiving and executing services on user terminals, which are often diverse and frequently lightweight, ranging from PCs to smartphones or pads to wearable devices.[10] The terminals need only store the underlying BIOS and a selection of protocols and management programs, thus enabling the terminals to be securely and easily managed and maintained.

Figure 3 shows a technical overview of the key technologies in transparent computing. There are eight technical points in a transparent computing platform, namely, multi-OS remote booting, streaming loading, lightweight virtualization, data transport protocols, server management, client interface, security, and performance optimization.

Representative Solutions for Implementing Transparent Computing

In the past decade, several studies have focused on transparent computing. Concomitantly, some systems based on its underlying concept have been successfully developed and invested for industrialization. In 2007, researchers developed a transparent computing system for PCs and proposed a novel Meta OS approach for streaming programs named 4VP.[10] In 2008, another team proposed a performance modeling and analysis algorithm for the Multi-OS booting process in a transparent computing environment.[11] In 2012, researchers developed a virtual machine-based network storage system for a transparent computing platform.[12] Since 2007, Intel has embedded UEFI into transparent computing architecture and developed many apps based on this combined platform.[5,6]

According to its definition, we can divide transparent computing's eight core technologies into three general ones, namely, transparent cloud architecture (Figure 3, parts 1 through 6), Meta OS (Figure 3, parts 1 through 3), and client implementation (Figure 3, part 6), plus two additional technologies (security, Figure 3, part 7, and performance optimization, Figure 3, part 8). The following subsections describe the three core categories in detail, followed by the additional technologies.

Transparent Cloud Architecture

In the transparent cloud architecture (see Figure 4), existing algorithms and schemes[13] that have been verified to improve cloud computing's performance can be employed at transparent servers to manage and schedule resources. The difference being that, in cloud computing, both data storage and computing are performed on servers, whereas in transparent computing, data storage happens on servers, but data computing happens on user terminals.[4]

A transparent computing system has two primary types of units: transparent servers and transparent users. Transparent servers are a cluster of cloud servers for storing OSs, apps, and user data. A well-developed system (Meta OS) runs on the server side and handles three jobs[14]: creating and managing an OS pool, creating and managing an app pool, and communicating with users based on transmission protocols. Additionally, management algorithms are employed on the server side to provide access control and file management.[13] In terms of transparent users on the user side, Meta OS is responsible for remote accessing, OS booting, and executing required apps. OSs and apps can run through lightweight virtualization technologies.[12]

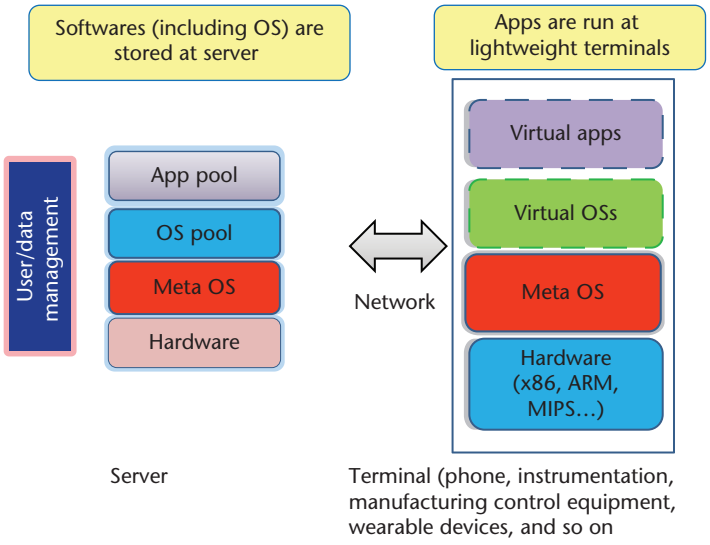

Figure 4. Transparent cloud architecture. In cloud computing, both data storage and computing are performed on servers, whereas in transparent computing, data storage happens on servers, but data computing happens on user terminals.

The transparent cloud architecture can support heterogeneous terminals, such as PCs, smartphones, and embedded and wearable devices. Figure 5 shows an example of heterogeneous terminal support. In this example, a Linux-supported calling service is stored on transparent servers and provided for user access. All OSs and apps stored on the servers are managed by Meta OS. Users need only install the Meta OS on their terminals; services, together with the supported OS (Linux), are transported to the terminals in a buffer-enabled block or streaming way.

As the foundation of transparent computing systems, the transparent cloud architecture has attracted significant attention from both research and industrial communities. Researchers have designed a virtual machine to implement a transparent computing system[15] that lets users enjoy services without needing to worry about technical issues. Based on that, other researchers developed a lightweight virtual machine, called LBTC, to implement a transparent computing system that can decrease extra overhead and redirect storage I/O requests to transparent servers.[16] Another study has proposed a virtual machine-based network storage system in which heterogeneous OSs and apps can be supported on demand without any modification.[12] A remote resource management method (TMON) has been proposed[17] for transparent computing, and a virtual machine monitor (VMM) has been exploited to enable client virtualization and computation with very little overhead, especially in support of graphically intensive apps.[18]

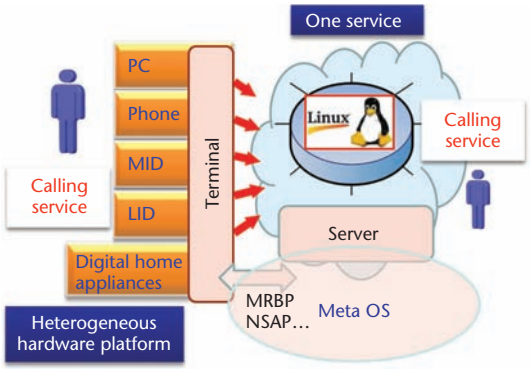

Figure 5. Example of heterogeneous terminal support in transparent computing. A Linux-supported calling service is stored on transparent servers and provided for user access. All OSs and apps stored on the servers are managed by Meta OS. Users need only install the Meta OS on their terminals.

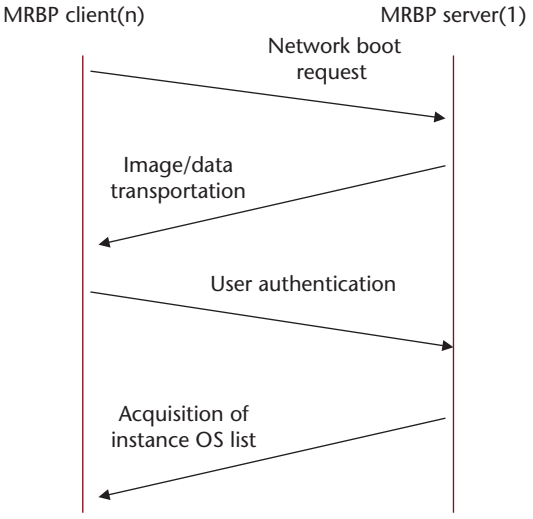

Figure 6. Interaction procedures of Multi-OS remote booting protocol (MRBP).

Meta OS

Located between the hardware platform and OS, Meta OS is responsible for managing hardware and software resources, including OSs, to provide users with secure and reliable computing services. The key technologies of Meta OS include kernel distribution, computing, storage, management separation, "block-flow" execution, cache, protocol, and virtualization. The Meta OS empowers users to select their desired services across different OS platforms, thus making computers more powerful and convenient. An improvement of Meta OS was proposed[7] to manage the separation of software and data, which systematically introduces the mechanisms and algorithms for separating software and user data.

Meta OS contains three critical components: Multi-OS remote booting protocol (MRBP), network service access protocol (NSAP), and virtual device, I/O, and resource management.

MRBP. The function of MRBP is to boot an OS remotely, from transparent servers to users' terminals, via networks based on a user's selection.[14] MRBP has two parts: an MRBP client running on the user side, and an MRBP server running on the server side. Figure 6 shows the interaction procedures of the MRBP protocol.

Some achievements related to MRBP have been proposed in the past decade. In 2003, researchers[19] presented a network-based client booting protocol, NCBP, which utilizes an extended dynamic host configuration protocol (DHCP) to obtain a local identifier. The proposed NCBP can load a batch script language environment based on a secure and active program transport protocol (APTP). In 2006, researchers[20] proposed a remote booting mechanism, in which the remote booting protocol is based on file downloading and can support multi-file OS kernel (Windows 2000 as well as XP, for example) booting protocols. Another study[8] proposed a new remote booting protocol, MRBP2, by which different OSs can be accessed from servers and dynamically executed on users' terminals. In 2008, researchers[11] designed a closed queuing network model to investigate the performance of remote booting in transparent computing for workload evaluation. In 2009, a study[21] proposed ENCBP, an extended NCBP for remotely booting multiple OSs, that can improve the performance of remote booting and further enhance security by differentiating the kernel types of loaded OSs.

NSAP. As an integral part of Meta OS, NSAP is employed to transmit signals, instructions, and data between transparent servers and users. In 2009, researchers[9] proposed NSAP for transparent computing, to provide reliable data transmission throughout a local area network (LAN) and realize OSs and other data shared among different users.

The working procedures of NSAP can be illustrated as follows. To start, NSAP establishes and maintains communication connections between transparent servers and users. Then, each user forms a request queue to store his or her service requirements and sends that queue to the servers. Upon receiving the user's request, servers analyze the request queue and read/write corresponding data from/to the storage

system. Finally, servers send a response, including the required data or services, to the user. In Meta OS, NSAP can collaborate with MRBP to complete data transmission and remote booting (see Figure 7).

Some related works have proposed performance improvements for data transmission in transparent computing. One group[22] significantly improved network file system (NFS) performance by setting the optimal block size for reading/writing and using TCP as an NFS transport protocol. Other researchers[23] proposed a dynamic trivial file transport protocol (DTFTP) that's based on a cross-layer design to improve data transmission performance and reduce packet loss. By utilizing DTFTP, the transmission rate can increase by 54 percent. Other studies[24,25] proposed an efficient multicast grouping mechanism over wireless local area network (WLAN), by which the average system loading time (average waiting time) can be greatly reduced and user experience can be improved.

Virtual device, I/O, and resource management. Virtual device and I/O management are of significant importance in transparent computing. In 2007, researchers[26] proposed an I/O management method, IOMan, to support multi-OS remote booting and apps running in LAN and to reduce the cost of using and maintaining computer systems. IOMan uses a software solution to set up disk access redirection; it doesn't need to modify the booting mechanism of certain commercial OSs, such as Windows, and doesn't affect other I/O operations. In 2008, researchers[27] proposed an optimal forecast algorithm and a prefetching strategy to accelerate I/O processing. In addition, researchers[28] proposed a novel method, Control-Splitted Data Transfer (CSDT), to process data transmission requests in transparent computing systems, which can improve the efficiency of data transmission.

In 2012, researchers[12] proposed a virtual machine-based network storage system for transparent computing to support heterogeneous OSs and apps. After that, they proposed a virtual storage device driver model for transparent computing, TVDSM, to support remote loading and running of heterogeneous OSs and apps on heterogeneous platforms.[29] A multimedia I/O access control policy for transparent computing, the classification and aggregation-based virtual I/O mechanism (CAVIO), was presented[30] to enhance multimedia I/O performance. In 2014, researchers[31] developed a virtual disk-based cloud computing platform, combining the characteristics of cloud computing and transparent computing, to provide heterogeneous services.

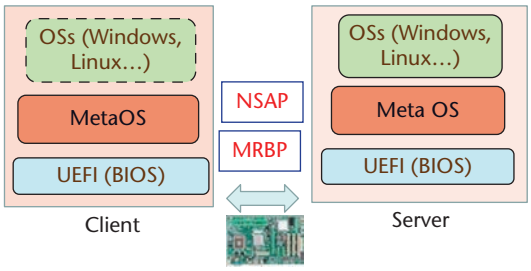

Figure 7. Collaboration of network service access protocol (NSAP) and MRBP.

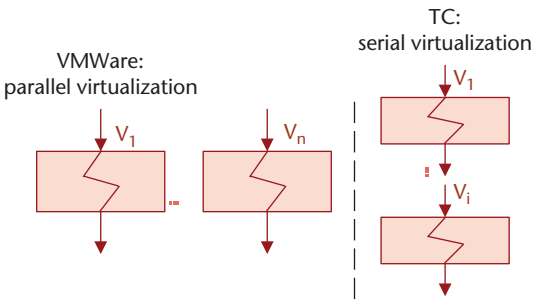

Figure 8. Virtualization differences between VMWare and transparent computing.

Importantly, virtualization in transparent computing is different from traditional virtualization technologies (such as VMWare). In VMWare, virtualization is parallel, while in transparent computing, virtualization is serially implemented from servers to users, which can be more adapted to lightweight terminals. Figure 8 shows the virtualization differences between VMWare and transparent computing are shown in Fig. 8.

In transparent computing, resource management mainly includes data and user management. In 2009, researchers[32] proposed a block-based data consistency method (BDCM) to solve the data inconsistency problem in transparent computing. It has a superior performance to normal PCs and similar systems with the same hardware. In 2013, a study[33] proposed a shared resource and service management (SRSM) scheme, in which heterogeneous resources and services can be dynamically scheduled and accounted on demand. In 2014, a researcher[34] designed a multi-OS booting program for smart terminals in mobile transparent computing that completes multi-OS booting by embedding lightweight system booting programs and Meta OS in smart terminals. In 2015, a team[35] presented a new solution to ensure quality of service for resource provisioning in virtualized environments.

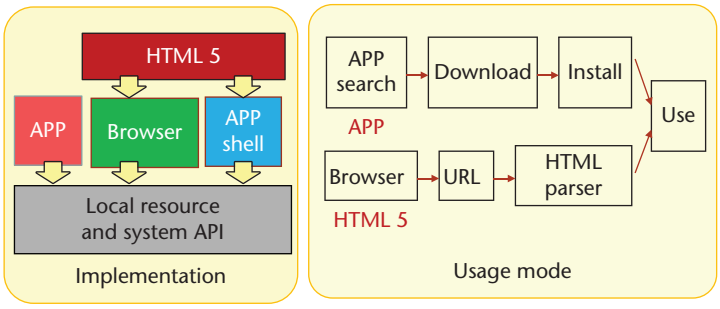

Figure 9. Differences between HTML 5 and traditional client apps.

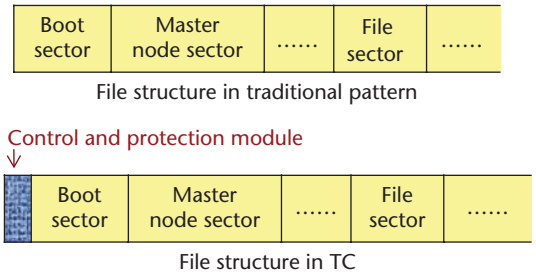

Figure 10. Control and protection module.

Client Implementation

One of the most important characteristics of transparent computing is its support of heterogeneous services over heterogeneous terminals. Therefore, client implementation is critical for transparent computing.

At the Intel Developer Forum 2012, representatives for Intel stated that HTML 5 is the preferred technology for achieving client implementation in transparent computing (http://download.intel.com/newsroom/kits/idf/2012_fall/pdfs/IDF2012_Renee_James.pdf). In October 2014, based on modifications to HTML 4 (standardized in 1997), the fifth revision of the HTML standard was proposed by the World Wide Web Consortium (W3C).[36] Compared with the previous versions, HTML 5 possesses the following new characteristics: rich semantic information, client canvas support, client offline storage support, rich multimedia support, cross-document message, and multithread support.

In transparent computing, HTML 5 is a suitable candidate for client implementation. Compared with traditional client apps, HTML5 has its own advantages (see Figure 9). HTML 5 possesses the advantages of cross-platform support and low development costs. However, browser differences and Web security attacks also pose challenges to the use of HTML 5. In transparent computing, a browser engine can be implemented down to the Meta OS layer, browser differences can be shielded, and cross-platform streaming execution can be achieved to improve the user experience.

Security

Although transparent computing can provide advanced security for transparent users, it's still vulnerable to some traditional security risks, such as privileged user access, viruses, malicious attack, and data theft. Meanwhile, this new computing paradigm also faces new security challenges,[37] including Multi-OS remote booting, virtual disk sharing, centralized security, confidentiality, integrity, and availability. To address the potential security risks in transparent computing systems, several approaches have been proposed.

One approach provides active protection in Meta OS.[38] Because Meta OS is located between the hardware and the OS layer, it can monitor software and user data. We can change the booting area of the file system and develop secure OS booting and data transmission protocols, such that security can be guaranteed in the OS booting process by adding a control and protection module over the original file system. Figure 10 describes the location of the control and protection module.

In 2007, researchers[38] proposed a security enhancement for MRBP and MRBP2, called RBIS (remote booting integrity service), to resist attacks by downloading code into the networks. Based on RBIS, transparent computing can guarantee a secure booting process based on integrity checksum, which can also establish credible links from UEFI firmware and pass up step-by-step to enhance security. Figure 11 shows the RBIS framework.

A second approach is to provide virus immunity via streaming execution. In transparent computing, all data obtained from transparent servers, including software and user data, will be eliminated from users' terminals after execution, such that viruses have few opportunities to jeopardize those terminals.

Another approach is to conveniently monitor and manage data streams. In transparent computing, monitoring points can be added to I/O devices for inspecting the transmitted data stream. Because data reading and writing can be intercepted by network interruption, data stream monitoring can ensure data integrity and accuracy (see Figure 12).

In 2010, researchers[37] summarized the security challenges in transparent computing. In 2014, a study[39] proposed the transparent computing security

architecture (TCSA), which empowers users to take the initiative to protect their own data. In addition, researchers[40] proposed a provably secure key agreement protocol using smart cards that's based on three-party authentication and can provide high-level security protection.

Performance Optimization

Performance optimization is another research focus in transparent computing. To improve quality of service and the overall user experience in transparent computing, existing works mainly optimize system performance from two aspects: buffer optimization and load balancing.

Buffer optimization involves designing buffer management algorithms to control the data buffers in both servers and user terminals.[41] On the user side, frequently accessed software and user data can be cached to reduce network load, while on the server side, frequently requested OSs, software, and user data can be pre-fetched from hard disks into memory with the purpose of reducing the I/O cost. In 2009, researchers[42] proposed a shared storage cache simulator (SSCS) algorithm that improves the traditional stack-distance model by developing user management and providing accurate cache performance analysis. In 2011, a study[43] presented the LRU-AFS algorithm based on the count threshold of data accessing times to increase the hit ratio. In 2012, researchers[44] proposed an optimal cache arrangement policy for transparent computing to improve system performance by analyzing workload characteristics and building a queuing model accordingly. In 2013, a team[45] designed a buffer adaptation method for high-speed data transmission. In 2015, researchers[46] proposed a TCCRM cache replacement algorithm, deployed on the server side, to improve the server cache hit rate and reduce the frequency of I/O responses.

Another method for improving transparent computing performance is to balance the data I/O requests among a cluster of servers. Researchers[47] proposed a dynamic load-balancing algorithm based on transparent computing to share resources among clusters of servers. Another team[48] theoretically formulated the data scheduling problem in transparent computing as a two-stage flow shop scheduling problem, which has been proven to be NP-hard. Accordingly, the researchers proposed two approximation algorithms to reduce the make span of responding data requests and theoretically analyzed the upper bounds of the proposed algorithms. Other works focused on developing load-balancing algorithms to improve energy efficiency[49] or investigated and optimized the performance of transparent computing from other aspects. One team[50] analyzed and evaluated the performance of LongXing Network Computers, which are developed based on transparent computing. Another work[51] proposed a simple algorithm to extend the traditional two-hop relay algorithms with delay constraints. Researchers[52] also proposed an enhanced algorithm to provide good performance in transparent computing systems by reducing the number of spurious timeouts and improving transmission efficiency.

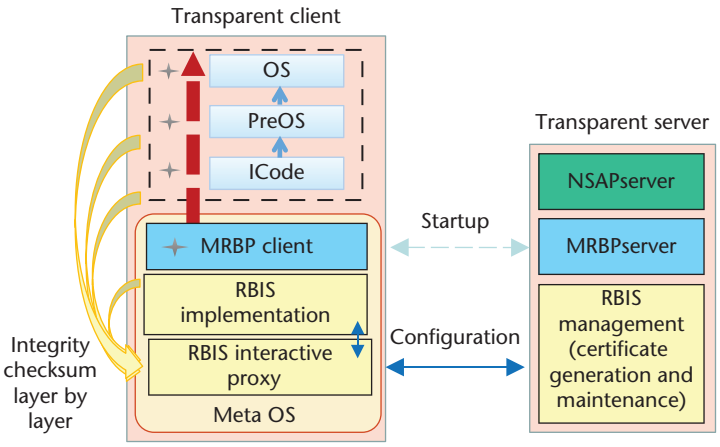

Figure 11. Remote booting integrity service (RBIS) framework.

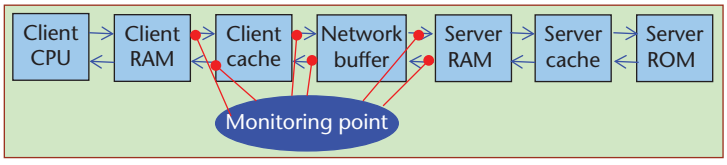

Figure 12. Monitoring and managing data streams.

Transparent Computing for Mobile Devices: New Challenges and Future Directions

Because of the inherent advantages of the computing paradigm, mobile transparent computing offers a promising way to make mobile devices more powerful, convenient, and secure. However, when applying this computing paradigm to mobile devices, we face some new challenges caused by new terminals and network environments.

New Challenges

The first new challenge is how to support lightweight and energy-constrained mobile user terminals in a transparent computing framework. Generally, desktops and laptops have more powerful processing

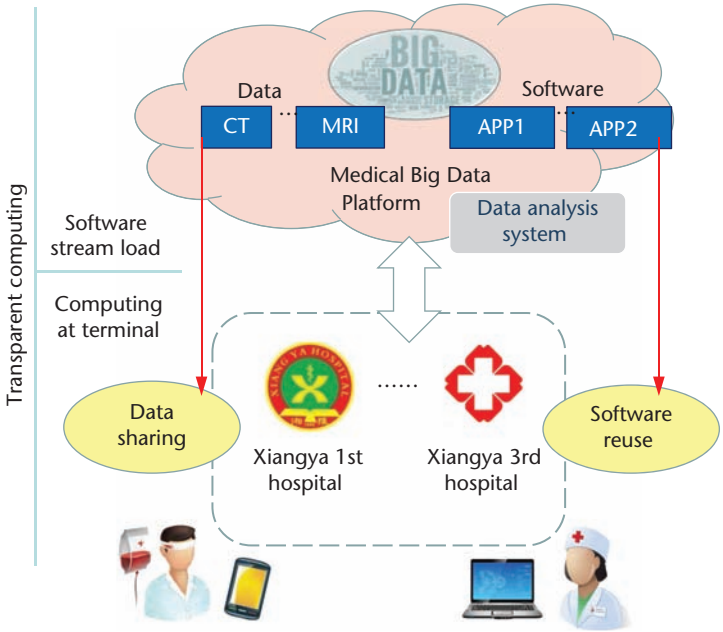

Figure 13. Xiangya medical big data platform.

capabilities than lightweight mobile terminals, such as mobile phones and wearable devices. Therefore, existing solutions for client implementation and resource management should be carefully improved for resource-limited mobile terminals. In 2013, researchers[53] proposed a completely cross-platform mobile transparent computing (CPMTC) method that can allocate on-demand resources and services to support various OSs and apps. Energy is a critical and unreplenishable resource in battery-powered mobile devices, thus, guaranteeing the battery life of terminals without compromising quality of service is an important and challenging research issue in mobile transparent computing.

Another challenge is the unstable and costly wireless communication found in the mobile Internet. Wireless networks are characterized by low-bandwidth, intermittent, and less-reliable transmission grounds, compared with wired networks. When mobile terminals connect to remote transparent servers to run apps, those apps and their users' data would be fully transmitted by unstable wireless communications. Although current wireless technologies, such as 4G and the anticipated 5G technology, might be able to facilitate high-speed data transmission, unstable wireless communications, caused by temporally and spatially varying wireless radio environments, still pose considerable challenges in guaranteeing the user experience in mobile transparent computing. Latency is another challenge that could have significantly adverse impacts in terms of energy efficiency and the interactive response of transparent computing applications by consuming abundant mobile resources and increasing transmission delays. Moreover, costly data traffic in the mobile Internet is a problem should also be addressed to encourage mobile transparent computing.

Last but not least is the challenge of supporting heterogeneous wireless networks and continuous user mobility while ensuring seamless connectivity to transparent servers. In mobile transparent computing, the integration of heterogeneous wireless networks (such as Wi-Fi and cellular networks) poses many technical challenges caused by the heterogeneity in different architectures, access protocols, and user mobility patterns. To address these challenges, it's critical to develop an adaptive protocol suit, similar to AdaptNet,[43] that can alleviate a plethora of heterogeneity issues, particularly rate adaptation and congestion control, and facilitate interoperation among various networks. Additionally, the integration and interoperation of heterogeneous networks in mobile transparent computing require lightweight, resource- and cost-effective, sustainable, and user-friendly approaches with optimized performance to address seamless mobility. A mobile user's movement out of range from transparent servers would prove fatal to the task being performed unless there are additional resource providers along the user's path, so it's important to maintain seamless connectivity with the least signal traffic and latency, to enrich the mobile user's experience.

Future Research and Development Directions

In academia, technical issues remain that require further investigation to improve the performance of mobile transparent computing. First, a tailored architecture should be designed for mobile transparent computing to alleviate the challenges caused by new terminals and network environments. The architecture should be able to jointly leverage the advantages of the transparent computing paradigm and the characteristics of mobile devices and networks to improve the mobile user's experience. Moreover, because mobile data traffic is tremendously hiked by ever-increasing user demands, future research efforts should be devoted to achieving high-speed and reliable data communication between transparent terminals and servers. In addition, because mobile devices have become the information hub in our daily lives, user data—including contextual information

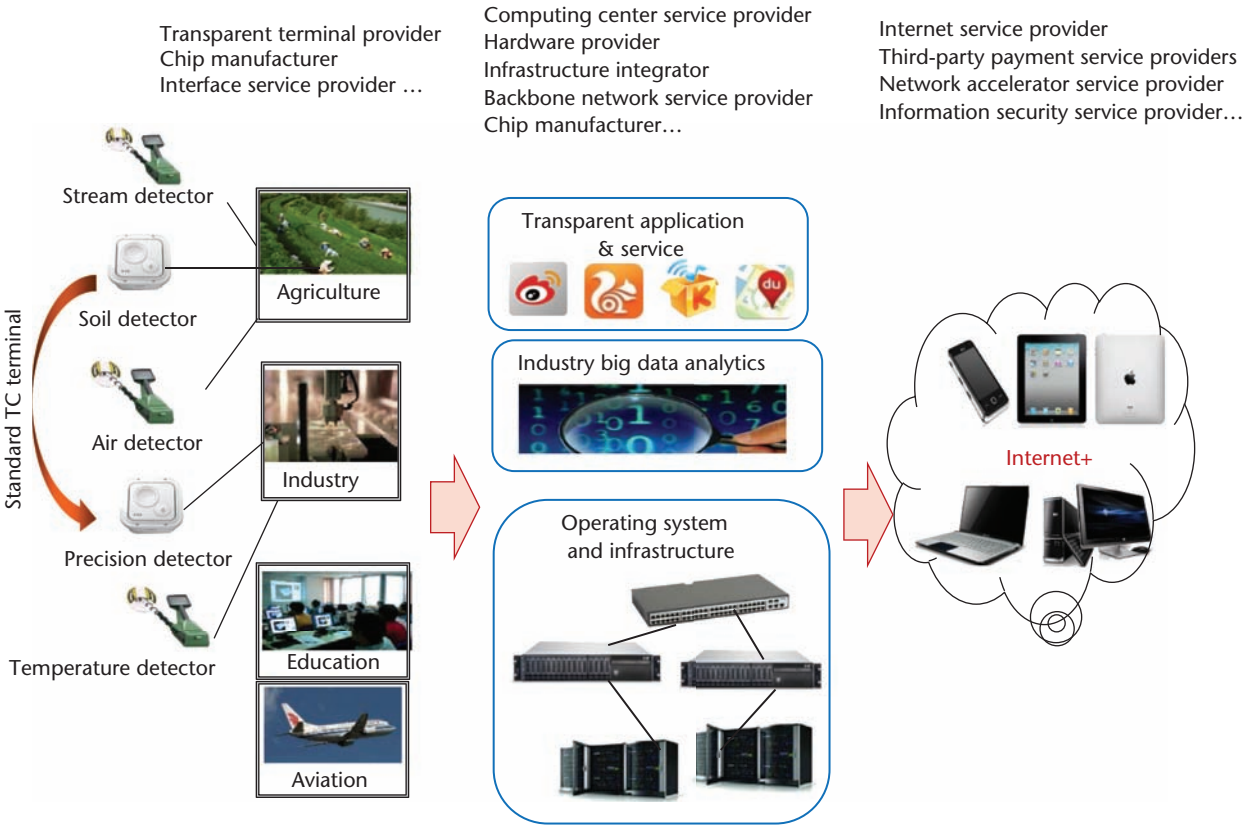

Figure 14. Some future applications of transparent computing.

such as location—can be leaked, putting user privacy into jeopardy. How to preserve user privacy in the mobile transparent computing framework thus constitutes a critical issue for future study.

In industry, transparent computing has experienced some successful application—for example, China Mobile's Aspire Company developed a new type of network operating system based on transparent computing, called TNOS. In Changsha, transparent computing is being applied to help build the Xiangya medical big data platform, which can support cross-regional, cross-hospital, and cross-system treatment services, medical information collection, and scientific research. In this platform, data and user privacy can be preserved without impacting health data access for mobile users (see Figure 13).

If the challenges of mobile transparent computing can be addressed effectively, transparent computing could create a new industrial chain from heterogeneous mobile and embedded terminals to new infrastructures and services. Figure 14 shows some future transparent computing applications, indicating that that transparent computing-based devices can be developed and deployed for different data collection and processing tasks. These data can then be transmitted to transparent severs, which can also be developed as a big data platform to perform industrial big data analysis.

Recently, a research team at the Central South University of China released a smart watch, TiWatch, which is being developed based on the concept of mobile transparent computing. Because smart watches are usually associated with smartphones, TiWatch uses the associated smartphone as a transparent server while the watch itself acts as a transparent client. The data communication between TiWatch and the smartphone is based on Bluetooth 4.1, and the watch's data computation is performed on demand in a page-streaming pattern. Based on this computing paradigm, TiWatch can run various apps smoothly with low-cost hardware and greatly improve energy efficiency. The battery life is reportedly longer than one week under normal usage.

As we step into the IoT era, where lightweight and mobile devices become dominate terminals in the Internet and our daily life, we're gaining

new opportunities, as well as new challenges, to exploit transparent computing from traditional terminals to mobile devices. Transparent computing is a promising computing paradigm in the current mobile computing era and will continue to attract increasing attention from both academia and industry.

Acknowledgments

This work is supported by the Major Science and Technology Research Program for Strategic Emerging Industry of Hunan (2012GK4106), International Science and Technology Cooperation Special Projects of China (2013DFB10070), Hunan Science and Technology Plan (2012RS4054), and Project of Innovation-driven Plan in Central South University (2015CXS010). We declare that we have no conflict of interests. For information on this article, please email Kehua Guo (guokehua@csu.edu.cn).

References

1. R. Branch et al., "Cloud Computing and Big Data: A Review of Current Service Models and Hardware Perspectives," *J. Software Eng. and Applications*, vol. 7, 2014, pp. 686–693.
2. Y. Zhang, "Transparence Computing: Concept, Architecture and Example," *Acta Electronica Sinica*, vol. 32, no. 12A, 2004, pp. 169–173.
3. Y. Zhang and Y. Zhou, "Transparent Computing: A New Paradigm for Pervasive Computing," *Ubiquitous Intelligence and Computing*, Springer, 2006, pp. 1–11.
4. Y. Zhou et al., "A Bare-Metal and Asymmetric Partitioning Approach to Client Virtualization," *IEEE Trans. Services Computing*, vol. 7, no. 1, 2014, pp. 40–53.
5. M. Wu, "Analysis and a Case Study of Transparent Computing Implementation with UEFI," *Int'l J. Cloud Computing*, vol. 1, no. 4, 2012, pp. 312–328.
6. Y. Zhang and Y. Zhou, "TransOS: A Transparent Computing Based Operating System for the Cloud," *Int'l J. Cloud Computing*, vol. 1, no. 4, 20121, pp. 287–301.
7. Y. Zhang, "The Challenges and Opportunities in Transparent Computing," *Proc. IEEE/IFIP EUC*, vol. 1, 2008, p. 3.
8. H. Yang et al., "MRBP2: A Transparence Computing Based Remote Booting Protocol," *J. Chinese Computer Systems*, vol. 27, no. 9, 2006, pp. 1657–1660.
9. W. Kuang et al., "NSAP: A Network Storage Access Protocol for Transparent Computing," *J. Tsinghua University Science and Technology*, vol. 49, no. 1, 2009, pp. 106–109.
10. Y. Zhang and Y. Zhou, "4VP: A Novel Meta OS Approach for Streaming Programs in Ubiquitous Computing," *Proc. IEEE AINA*, 2007, pp. 394–403.
11. G. Guo et al., "Performance Modeling and Analysis of the Booting Process in a Transparent Computing Environment," *Proc. IEEE FGCN*, vol. 1, 2008, pp. 83–88.
12. Y. Gao, Y. Zhang, and Y. Zhou, "Building a Virtual Machine Based Network Storage System for Transparent Computing," *Proc. IEEE CSSS*, 2012, pp. 2341–2344.
13. Y. Zhang and Y. Zhou, "Transparent Computing: Spatio-Temporal Extension on Von Neumann Architecture for Cloud Services," *J. Tsinghua University Science and Technology*, vol. 18, no. 1, 2013, pp. 10–21.
14. Y. Zhou and Y. Zhang, *Transparent Computing: Concepts, Architecture, and Implementation*, Cengage Learning Asia, 2010.
15. G. Xu et al., "Design and Implementation of a Virtual Machine-Based Transparent Computing System," *J. Tsinghua University Science and Technology*, vol. 48, no. 10, 2008, pp. 1675–1678.
16. C. Chen et al., "Lightweight Virtual Machine-Based Transparent Computing System," *Computer Eng.*, vol. 36, no. 11, 2010, pp. 39–40.
17. Y. Gao, Y. Zhang, and Y. Zhou, "A Remote Resource Management Method for Transparent Computing," *Proc. IEEE CSIP*, 2012, pp. 1378–1381.
18. Y. Zhou, "Transparent Computing: From Concept to Implementation," *Proc. IEEE ITME*, vol. 1, 2012, p. 5.
19. Y. Zhou, Y. Zhang, and Y. Wang, "A Customizable Boot Protocol for Network Computing," *J. Software*, vol. 14, no. 3, 2003, pp. 538–546.
20. X. Wang, N. Xia, and H. Yang, "A Network Storage Protocol Based Operating System Remote Boot Mechanism," *Computer Eng. & Applications*, vol. 42, no. 20, 2006, pp. 95–98.
21. W. Li et al., "ENCBP: An Extended Multi-OSes Remote Booting Method," *J. Computer Research and Development*, vol. 46, no. 6, 2009, pp. 905–912.
22. S. Guo, W. Yang, and G. Wang, "NFS Protocol Performance Analysis and Improvement for Mobile Transparent Computing," *Proc. IEEE HPCC*, 2013, pp. 1878–1883.
23. J. Li, "File Transfer Protocol Research in Mobile Transparent Computing Based on Cross-Layer Design," master's thesis, Central South Univ., 2014.
24. H. Yin, W. Yang, and G. Wang, "Wireless Multicast for Mobile Transparent Computing," *Proc. IEEE HPCC*, 2013, pp. 1884–1889.
25. H. Yin, "Wireless Multicast for Mobile Transparent Computing in WLANs," master's thesis, Central South Univ., 2014.
26. N. Xia et al., "IOMan: An I/O Management Method Supporting Multi-OS Remote Boot and Running," *Jisuanji Yanjiu yu Fazhan (Computer Research and Development)*, vol. 44, no. 2, 2007, pp. 317–325.

27. H. Tan, Y. Wang, and J. Sim, "A Dynamic Rebuild Strategy of Multi-Host System Volume on Transparence Computing Mode," *Proc. IEEE HPCC*, 2008, pp. 981–986.
28. G. Xu, W. Kuang, and Y. Zhou, "CSDT: Method for Data Transfer between Client Devices of Transparent Computing Systems," *Computer Eng.*, vol. 34, no. 17, 2008, pp. 1–3.
29. Y. Gao, Y. Zhang, and Y. Zhou, "TVSDM: A Virtual Storage Device Driver Model for Transparent Computing," *J. Tsinghua University Science and Technology*, vol. 53, no. 7, 2013, pp. 1064–1068.
30. Y. Gao, Y. Zhang, and Y. Zhou, "A Multimedia I/O Access Control Policy for Transparent Computing," *J. Hunan University Natural Sciences*, vol. 40, no. 3, 2013, pp. 93–98.
31. Y. Zhou et al., "TransCom: A Virtual Disk-Based Cloud Computing Platform for Heterogeneous Services," *IEEE Trans. Network & Service Management*, vol. 11, no. 1, 2014, pp. 46–59.
32. G. Guo and Y. Zhou, "Block-Based Data Consistency Method for Transparent Computing," *J. Tsinghua University Science and Technology*, vol. 49, no. 10, 2009, pp. 1721–1724.
33. Y. Xiong, S. Huang, and M. Wu, "Shared Resource and Service Management for Mobile Transparent Computing," *Proc. IEEE HPCC*, 2013, pp. 1846–1853.
34. S. Guo, "Multi-OS Boot for Smart Terminals and Its Performance Analysis and Optimization in Mobile Transparent Computing," master's thesis, Central South University, 2014.
35. J. Liu et al., "Aggressive Resource Provisioning for Ensuring QoS in Virtualized Environments," *IEEE Trans. Cloud Computing*, vol. 3, no. 1, 2014, p. 1.
36. G. Anthes, "HTML5 Leads a Web Revolution," *Comm. ACM*, vol. 55, no. 7, 2012, pp. 16–17.
37. Y. Zhang et al., "Information Security Underlying Transparent Computing: Impacts, Visions and Challenges," *Web Intelligence & Agent Systems*, vol. 8, no. 2, 2010, pp. 203–217.
38. W. Kuang et al., "RBIS: Security Enhancement for MRBP and MRBP2 Using Integrity Check," *J. Chinese Computer Systems*, vol. 28, no. 2, 2007, pp. 251–254.
39. G. Wang et al., "Security from the Transparent Computing Aspect," *Proc. Int'l Conf. Computing, Networking and Communications*, 2014, pp. 216–220.
40. H. Yang et al., "Provably Secure Three-Party Authenticated Key Agreement Protocol Using Smart Cards," *Computer Networks*, vol. 58, no. 1, 2014, p. 2938.
41. Y. Fang et al., "On Delivery Delay-Constrained Throughput and End-to-End Delay in MANETs," *Proc. IEEE HPCC*, 2014, pp. 456–463.
42. L. Wei, Y. Zhang, and Y. Zhou, "Simulation Analysis and Validation of Cache Performance in TransCom Systems," *J. Tsinghua University Science and Technology*, vol. 49, no. 10, 2009, pp. 1700–1703.
43. C. Tan et al., "Transparent Computing System Based on Hierarchical Cache," *Computer Eng.*, vol. 37, no. 5, 2011, pp. 270–272.
44. Y. Gao, Y. Zhang, and Y. Zhou, "Performance Analysis of Virtual Disk System for Transparent Computing," *Proc. IEEE UIC/ATC*, 2012, pp. 470–477.
45. H. Liu et al., "Receiving Buffer Adaptation for High-Speed Data Transfer," *IEEE Trans. Computers*, vol. 62, no. 11, 2013, pp. 2278–2291.
46. Y. Xu, "Optimization of Transparent Computing System Based on Virtualization," master's thesis, Central South University, 2015.
47. H. Yang et al., "A Dynamic Load Balancing Algorithm Based on Transparence Computing," *Computer Eng.*, vol. 32, no. 13, 2006, pp. 133–135.
48. J. Ren, Y. Zhang, and J. Chen, "Analysis on the Scheduling Problem in Transparent Computing," *Proc. IEEE HPCC*, 2013, pp. 1832–1837.
49. D. Zhang et al., "Leveraging the Tail Time for Saving Energy in Cellular Networks," *IEEE Trans. Mobile Computing*, vol. 13, no. 7, 2014, pp. 1536–1549.
50. G. Guo, Y. Zhang, and W. Li, "Performance Report of Longxing Network Computer Based on Transparent Computing," *Computer Eng. and Applications*, vol. 41, no. 17, 2005, pp. 19–21.
51. Y. Fang et al., "A Delay Constrained Two-Hop Relay Algorithm for Transparent Computing in MANETs," *Proc. IEEE HPCC*, 2013, pp. 2337–2341.
52. X. Wang, N. Xia, and L. Wei, "RTO Algorithm for Transparent Computing Systems," *J. Tsinghua University Science and Technology*, vol. 47, no. 10, 2007, pp. 1696–1699.
53. W. Liang, Y. Xiong, and M. Wu, "A Cross Platform Computing Method and Its Application for Mobile Device in Transparent Computing," *Proc. IEEE HPCC*, 2013, pp. 1838–1845.

Yaoxue Zhang is a fellow of the Chinese Academy of Engineering and a professor in computer science and technology at Central South University, China. His research areas include computer networking, operating systems, ubiquitous/pervasive computing, transparent computing, and active services. Zhang received a PhD in computer networking from Tohoku University. Contact him at zxy@csu.edu.cn.

Kehua Guo is an associate professor in the School of Information Science and Engineering, Central South

University. His research interests include social computing, ubiquitous computing, big data, and image retrieval. Guo received a PhD in computer science and technology from the Nanjing University of Science and Technology. Contact him at guokehua@csu.edu.cn.

Ju Ren is a PhD candidate in the Department of Computer Science at Central South University, China. His research interests include wireless sensor network, mobile sensing/computing, and cloud computing. Ren received an MS in computer science from Central South University, China. Contact him at ren_ju@csu.edu.cn.

Yuezhi Zhou is an associate professor in the Department of Computer Science at Tsinghua University, China. His research interests include cloud computing, distributed systems, mobile devices, and systems. Zhou received a PhD in computer science and technology from Tsinghua University, China. He received the IEEE Best Paper Award at the IEEE AINA International Conference in 2007. He's a member of IEEE and ACM. Contact him at zhouyz@mail.tsinghua.edu.cn.

Jianxin Wang is the vice dean and a professor in the School of Information Science and Engineering, Central South University. His research interests include algorithm analysis and optimization, parameterized algorithm, and computer networks. Wang a PhD in computer science from Central South University, China. He's a senior member of IEEE. Contact him at jxwang@csu.edu.cn.

Jianer Chen is a professor of computer science at Texas A&M University and Central South University in Changsha, China. His research is centered on computer algorithms and their applications. Chen received a PhD in computer science from Courant Institute of New York University and in mathematics from Columbia University. Contact him at jianer@csu.edu.cn.