Wenjun Zeng
University of Missouri-Columbia

Virtualized Screen: A Third Element for Cloud–Mobile Convergence

Yan Lu, Shipeng Li, and Huifeng Shen
Microsoft Research Asia

In recent years, we have witnessed the strong growth of cloud computing, which refers to both applications delivered as services over the Internet and the infrastructure (including both hardware and software systems) in the data centers that supports those services.[1] In general, cloud computing provides a centralized platform where programs can be executed and data can be stored. Cloud services built on the platform can offload many computing tasks normally solely done on the client devices and can be accessed instantly from anywhere at any time. However, until recently, the place where screen images that interface with users are rendered has been mostly done locally on client devices.

Our argument is that the screen rendering can also be moved to the cloud and the rendered screen can be delivered as part of the cloud services. In general, the screen represents the whole or part of the display images. In a broad sense, it also represents a collection of data involved in user interfaces such as display images, audio data, mouse, keyboard, pen and touch inputs, and other multimodality inputs and outputs. In this article, we use the term *screen* to refer to the display images. There are a number of advantages to putting screen rendering in the cloud.

First, because screen rendering is closely coupled with program execution and data storage, putting screen rendering in the cloud where data storage and program execution are centralized actually simplifies the cloud computing architecture. Second, screen rendering, especially graphics-rich screen rendering, is not a trivial task. It often requires a powerful CPU and GPU on the client devices. Moving screen rendering to the cloud will greatly reduce the hardware requirement for client devices and thus make low-cost devices possible. Third, even if the client devices are equipped with powerful CPU and GPU processors, offloading all or part of screen-rending task to the cloud will spare the client processing power to handle more effectively higher priority tasks (such as the local UI) and rich interactions (such as touch and gesture recognition) that require fast responses locally. Last, screen rendering in the cloud also brings a new avenue for optimizing the overall computing experience.

Screen virtualization or screen rendering in the cloud doesn't always mean putting the entire screen-rendering task in the cloud. Depending on the actual situations—such as local processing power, bandwidth and delay of the network, data dependency and data traffic, and display resolution—screen rendering can be partially done in the cloud and partially done at the clients (that is, scalable screen virtualization); collaboratively, a complete rendered screen is presented to the user. This is very similar to traditional cloud computing in which we have to decide where we should put the program execution and data storage, either remotely in the cloud or locally on the device, to obtain an optimized computing experience.

Editor's Note

Mobile and cloud computing have emerged as the new computing platforms and are converging into a powerful cloud–mobile computing platform. This article envisions a virtualized screen as a new dimension in such a platform to further optimize the overall computing experience for users. In a virtualized screen, screen rendering is done in the cloud, and delivered as images to the client for interactive display. This enables thin-client mobile devices to enjoy many computationally intensive and graphically rich services. Technical challenges are discussed and addressed. Two novel cloud-mobile applications, Cloud Browser and Cloud Phone, are presented to demonstrate the advantages of such a virtualized screen.

—Wenjun Zeng

1070-986X/11/$26.00 © 2011 IEEE

Published by the IEEE Computer Society

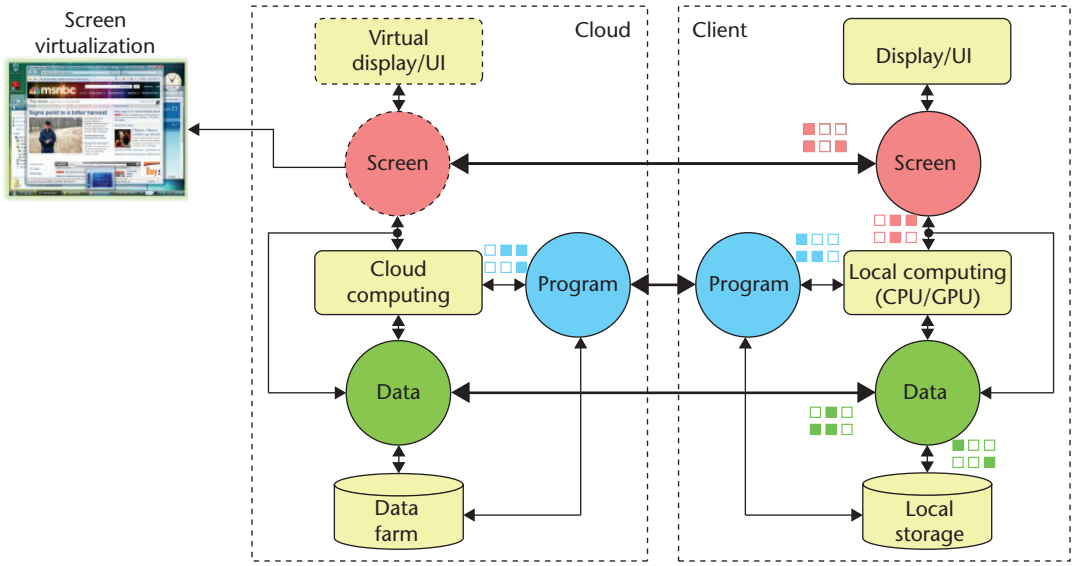

Figure 1. The conceptual diagram of the cloud–client computing architecture.

This flexibility added by screen virtualization can enable us to provide an even further optimized computing experience by balancing between local client devices and remote clouds in terms of data storage, program execution, and now, screen rendering.

However, rendering a screen in the cloud also introduces obstacles for the client devices to access the virtual screen, if it needs to maintain high-fidelity display images and responsive user interactions. Fortunately, we have already developed a number of advanced multimedia and networking technologies to address these issues. Ultimately, we would like to define a common cloud API for cloud computing with scalable screen virtualization, with which the developers never have to care where the data storage, program execution, and screen rendering actually occur because the cloud services for the API will adaptively and optimally distribute the storage, execution, and rending among the cloud and the clients. Similar to the GUI, which drove the wide grassroots adoption of personal computers, the cloud API, which bridges the cloud and a wide variety of clients, will introduce new computing experiences both locally and remotely. It will drive the evolution of cloud–mobile computing into a revolution.

Screen in the cloud

Rapid development of the Internet has provided opportunities for using remote computing and storage resources hosted by powerful, parallel, distributed machines in a public or private data center. In a typical cloud-client computing architecture, the data and the program can be stored, loaded, and run remotely and/or locally. To take advantage of the cloud, computationally intensive tasks are usually undertaken in the cloud to generate some intermediate results (for example, HTML data), which are then delivered to the clients for the creation of a locally processed display screen. In other words, local screen rendering is separated from the data storage and program execution, and is connected to them through the Internet.

Up until now, the virtualization of the screen in the cloud has been an underdeveloped area in cloud computing. An even less addressed area is how to leverage the virtual screen in the cloud and combine it with local rendering capabilities to give the same or even better user experiences across different devices, regardless of their computing capability, rendering capability, bandwidth, and screen resolution. Figure 1 depicts the proposed conceptual architecture for cloud–client computing, in which the virtual screen is rendered in the cloud. Behaving like the data and the program, the screen can be processed adaptively and collaboratively between the cloud and the clients, determined by the client and cloud capabilities. As we mentioned earlier, the cloud–client architecture with screen adaptation could provide the following benefits:

- an optimized interactive experience to end users across different screens, such as laptop, television, and phone;

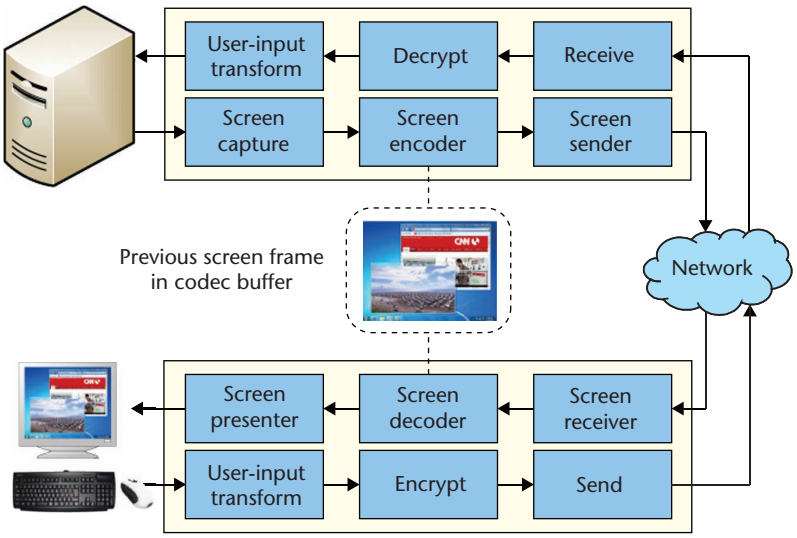

Figure 2. The interactive screen-remoting system.

- a simplified programing model for developers, as if they were building local applications without struggling on various types of real-time data communications; and
- a quick model for software providers to deploy software services in the cloud, through a mechanism of virtual machines plus screen remoting—for example, this model could help to quickly deploy trial software without piracy concerns.

Is it feasible to deliver real-time screen as services through the Internet to achieve near-local interactive experiences? The screen as display images indeed contains a large volume of data. Fortunately, its representation as image sequences has a lot of redundancy that can be efficiently removed. Moreover, screen refreshment with a full-frame unit can make the latency of screen updating at the client side relatively constant. While breakthroughs of multimedia, hardware, and networking technologies are around the corner, our recent progress on a remote-computing system with virtualized screen implies that screen compression and transmission can be very efficient even with technologies available today.

Remote computing with virtualized screen

Following the cloud-computing conceptual architecture depicted in Figure 1, we have developed a thin-client, remote-computing system that leverages interactive screen-remoting technologies.

System architecture

Thin-client, remote-computing systems are expected to provide high-fidelity displays and responsive interactions to end users as if they were using local machines. However, the complicated graphical interfaces and multimedia applications usually present technical challenges to thin-client developers for achieving efficient transmissions with relatively low-bandwidth links. Figure 2 depicts the proposed thin-client, remote-computing system, which decouples the application logic (remote) and the user interface (local) for clients to use remote servers deployed as virtual machines in the cloud. The servers and the clients communicate with each other over a network through an interactive screen-remoting mechanism. The clients send user inputs to the remote servers, and the servers return screen updates to the clients as a response.

The screen-update model determines whether the screen can be efficiently compressed and transmitted to the client. Most of the existing thin-client systems, such as Virtual Network Computing (VNC)[2] and Remote Desktop Protocol (RDP),[3] represent screen updates with graphics primitives of arbitrarily-sized regions. Such a mechanism allows the server to simply forward the graphics primitives to be updated into the compressors and discard other stable regions directly. At the client side, the screen presenter renders the received graphics primitives and overlays rectangular areas of pixels in destined regions. However, the regions to be updated are usually small and at arbitrary positions, such as menus or edit boxes. Encoding these small regions with arbitrary locations will result in the system suffering from compression efficiency degradation.

In contrast to the architecture based on arbitrarily-sized regions, our thin-client system employs a frame-based screen-representation model. This model reads all the pixels on the screen from the frame buffer at once and feeds the whole screen image into the compressors and transmitters. At the client side, the screen presenter replaces the whole screen with the newly decoded one. As shown in Figure 2, both the server and the client store the same reference frame, which is used to remove the redundancies between consecutive frames. Besides the advantage in compression, the frame-based representation model also simplifies the system architecture without the

overhead in scheduling the updates of screen regions. Furthermore, this frame-based representation model could recover from errors due to packet loss very quickly without retransmissions through keyframe refreshing.

Compared with existing remote-computing systems, such as VNC and RDP, the proposed solution could significantly improve user experience in terms of both the smoothness of the screen update and the response in interaction, which are enabled through the advanced screen-compression and transmission technologies introduced next.

Screen compression

Screen images might include, but are not limited to, webpages, slides, posters, images, videos, and anything showing up on your computer screen. For natural pictures, many existing image- and video-coding standards (such as, JPEG2000 and H.264/Advanced Video Coding) have demonstrated excellent coding performances. However, they are inefficient for compressing screen images that in most cases contain rich text. The challenges for compressing screen images are summarized as follows:

- *Computational complexity*. The screen codec has to handle a large amount of data to maintain high resolution and high-frame-rate screen updates. Meanwhile, the codec also has to spare most of the processing resources for other regular applications.

- *Compression performance*. The text, graphics, and pictures composing a screen image each possess a unique set of characteristics and susceptibilities to encoding artifacts. The screen codec has to handle them efficiently within a single framework, and yet maintain low computational cost.

In general, a typical screen image can be segmented into four categories of regions: smooth, text, picture, and text-on-picture. Based on our statistical analysis, the smooth and picture regions are more suitable for coding in the transform domain, while the text and text-on-picture regions can be efficiently compressed in the pixel domain. To simplify the codec architecture, we propose a block-based coding algorithm with only two coding types: picture and text. The text-block coding scheme also includes an escape pixel-coding technique, which is specifically designed to handle pixels belonging to a background picture in a text-on-picture region.

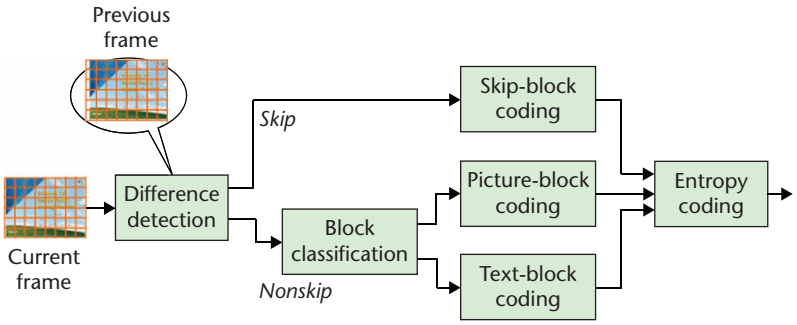

Figure 3. Block diagram of the proposed screen compression algorithm.

Figure 3 depicts the proposed screen-compression algorithm. The captured screen image is segmented into nonoverlapping 16×16 blocks. After comparison with the previous frame, each block is identified as either a skip or a nonskip block. Further, a classification is performed to separate the nonskipped blocks into picture blocks and text blocks, followed by the adaptive-coding schemes associated with the block types. The statistical gradient histogram is employed for the purpose of block classification. Obviously, 100 percent accurate block classification is almost impossible because of the diversity of text-on-picture regions. Fortunately, the proposed escape pixel-coding scheme can avoid quality loss with a marginal rate increase in case of incorrect classification.

This screen codec outperforms traditional natural image and video codecs for coding screen images in terms of rate, quality, and computational complexity. The encoding and decoding processes are also GPU-friendly in implementation. GPU acceleration with a mainstream video card could easily double the frame rates of screen capturing and coding over systems without GPU acceleration.

Screen transmission

The latency of screen transmission is probably the most important factor to user experience in a remote-computing system. Because the screen images are organized as a time-series-like a video, some existing video-transmission technologies could be leveraged here. The strong dependency between predictively coded frames makes the video stream sensitive to transmission errors. Therefore, some buffering and error-control mechanisms have to be employed, which causes additional delay on

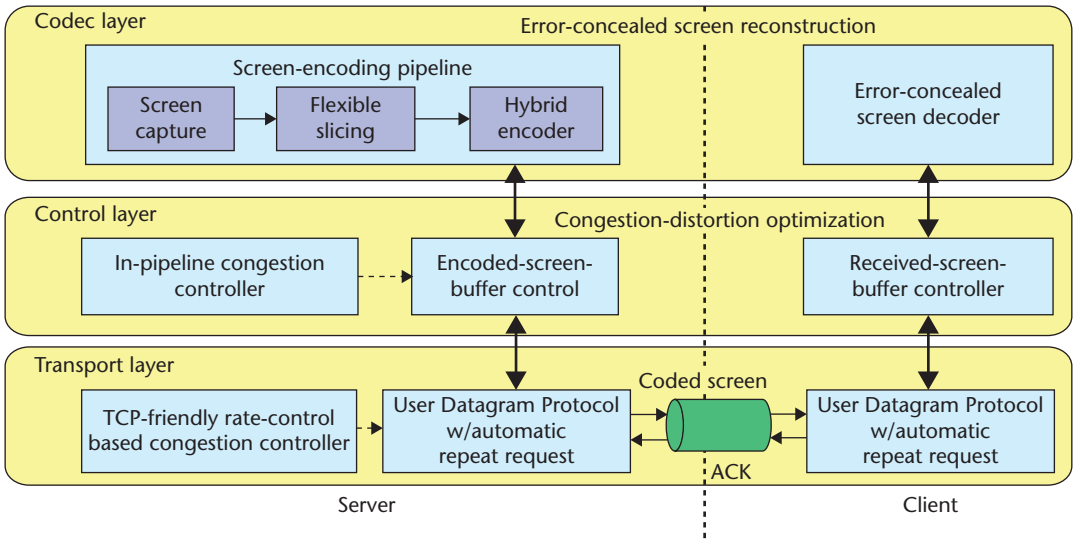

Figure 4. Block diagram of the proposed screen transmission algorithm.

top of the inherent network-transmission delay.

Furthermore, interactive screen remoting has much more stringent requirements on latency than real-time video communications. For example, the user usually expects an immediate response on the local display after clicking a button, similar to what would be expected when using a local machine. Achieving this performance requires quick round-trip message processing and almost instant updates between the virtual screen in the cloud and the local display on the device. Therefore, existing video-transmission technologies may not be sufficient to the interactive screen-remoting scenario.

Fortunately, the proposed screen codec doesn't introduce strong dependency between consecutive frames, which can mitigate to a certain extent the drifting errors in decoded screen images caused by transmission errors. Moreover, screen remoting presents some unique features different from video-transmission scenarios. For example, it's not always necessary to update the whole screen frame all at once. Rather, some out-of-order screen regions can be presented early as they become available. Therefore, it's possible to achieve extremely-low-latency transmission with the help of some content-aware scheduling schemes.

Figure 4 illustrates the proposed screen-transmission algorithm, which is based on the joint optimization of screen compression and transmission. The screen images are processed in three layers:

- At the codec layer, the screen images are encoded with a flexible slicing structure to help error-concealed screen reconstruction at the client. Moreover, the screen decoder can handle out-of-order screen packets without producing drifting errors.
- At the control layer, the congestion-distortion optimization algorithm schedules the packets in the sending buffer. These packets are scheduled on the basis of their impact on network congestion and display distortion. Some packets may be dropped without reencoding the frames followed.
- At the transport layer, the User Datagram Protocol transport with automatic repeat request is employed to balance the transmission rate and error control. To avoid congestion in the network, TCP-friendly rate control is used to estimate available bandwidth.

Cloud-mobility convergence

The rapid evolution of mobile computing offers a wide variety of freedom to mobile users. Besides communication functions, a mobile device could be a computing, sensor, control, gaming, and natural-interaction platform. However, it's difficult for a mobile device to serve as a dominant device for all the user's computing needs, mainly because of its limited capabilities in terms of computing, storage, display, and interaction. On the other hand, cloud computing offers unlimited computing and storage capabilities through centralized data centers. The capacity of mobile devices

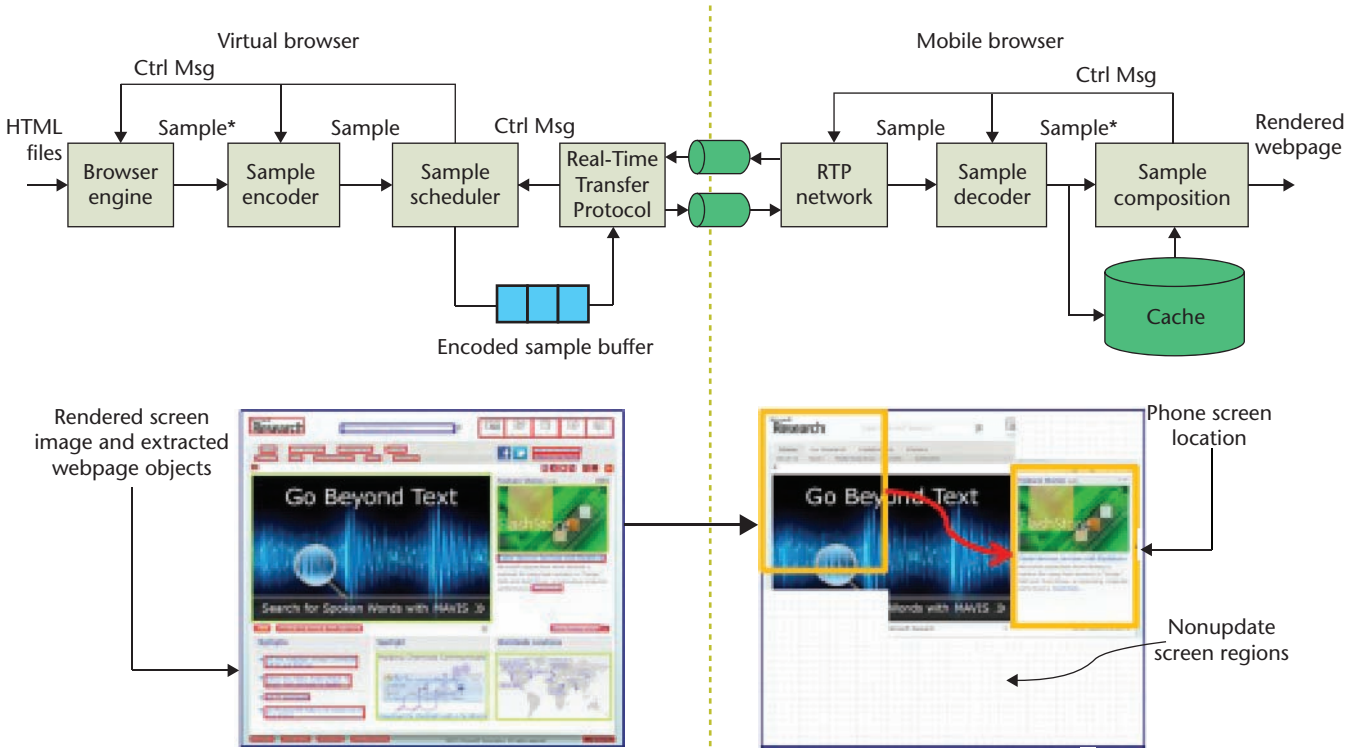

Figure 5. The system architecture of the Cloud Browser.

could be significantly augmented by cloud computing:

- *Remote-computing terminal.* A mobile device with remote-desktop solutions (for example, RDP and VNC) could use remote computer programs in the cloud, though existing computer programs are usually designed for a relatively large display and difficult to adapt to mobile screens.

- *Cloud-services harnessing platform.* A mobile device could access unlimited services in the cloud through the Internet. In an extreme case, except for the user interfaces and sensors, all program execution, data storage, and screen rendering could be done in the cloud. In general, by balancing program execution, data storage, and screen rendering between the cloud and the mobile device, we could achieve an optimized computing experience adaptive to the capacities of the cloud and the mobile device.

The virtualization of the screen in the cloud introduces a new avenue to further optimize the cloud–mobile computing experience and could potentially accelerate cloud–mobile convergence. On the basis of the proposed interactive screen-remoting system, we showcase two exemplar mobile applications: Cloud Browser and Cloud Phone.

Cloud Browser

As most mobile phones come with preinstalled Web browsers, mobile browsing is becoming increasingly popular. However, many mobile browsers have limited support for complicated HTML objects, such as full-feature Flash, JavaScript, and Silverlight. Installation of add-ons from some websites in many cases isn't possible. Even worse, the website owners might have to develop and maintain two versions of the same content, one full version for desktop browsers, and the other simplified version for mobile browsers. In most cases, the mobile version loses rich multimedia information and results in a compromised and degraded browsing experience. It's desirable to bring the desktop-like browsing experiences to mobile phones.

Recently, we have built a cloud-assisted mobile browser, the Cloud Browser, which is an extension to the proxy-based Web browser we developed.[4] Figure 5 illustrates the architecture of the Cloud Browser, which consists of a virtual browser deployed in the cloud and a mobile browser installed on the device. The virtual browser extracts and renders webpage objects into samples, encodes and multiplexes them

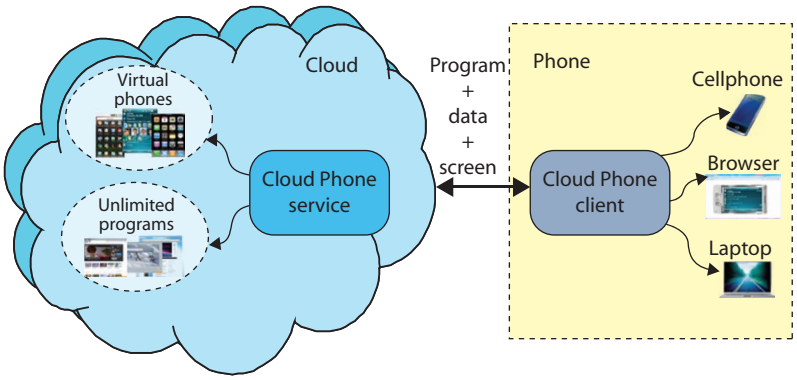

Figure 6. Conceptual diagram of the Cloud Phone.

into a composite screen format, and then sends the data to the client with optimized scheduling. The mobile browser receives and decodes these samples, and assembles them for presentation to mobile users. The mobile browser also sends back control messages, such as visibility information about the client display and user-interaction events, to the virtual browser.

As described previously, the composite screen format is defined as a sequence of samples, where the sample can be either a screen image or a webpage object. The webpage object consists of object ID, type, and properties such as the position in a screen. We define four types of webpage objects: hyperlink, text input, password input, and dynamic. The object samples are not intended to be used for direct display, but for improving user interaction experiences and avoiding unnecessary bandwidth consumption. For example, the client can use the object information to execute some simple interaction logic to give quick responses to the user. Moreover, all or part of the screen could be selectively updated according to the object types and the user's options.

The Cloud Browser is a complete end-to-end solution that offloads the rendering of webpages from the client to the cloud or some proxy servers, reformats the webpages to eliminate incompatibility with mobile devices, and maximizes the object-level interactivity while keeping the full desktop browsing experience. In many cases, the webpage's bits after compression could be even less than that of the original HTML files. Even better, the proposed mobile browser could significantly reduce page loading time by pulling only a visible portion of the screen image rendered in the cloud for that webpage. Moreover, because most screen samples have already been rendered in the cloud, the local client could quickly assemble these samples into a webpage for the users to view to further shorten the webpage rendering time. While the HTML webpage could be converted to a screen image, the screen image could also be embedded into the HTML file as a webpage element to help real-time cloud-client interactive communications.

Cloud Phone

The smartphones as a personal computing device are broadly available to mobile users. In addition to a wide variety of phone brands and a number of different operating systems in the market, the rapid development of hardware and application software accelerates the pace at which phone hardware is upgraded. Is it possible to replace the frequent hardware upgrades with only software and services upgrades? Recently, we have built a prototype of a complete cloud-service-based phone solution, the Cloud Phone, with which the phone user can access the ever increasing programs and services available in the cloud, even on a low-cost device without the need to update phone hardware anymore.

Figure 6 illustrates the conceptual diagram of the Cloud Phone. On the one side, with the virtual phones deployed in the cloud with user-preferred mobile operating systems (for example, Windows Phone, iPhone, and Android) and the unlimited apps or services, the Cloud Phone service could switch between these phone user interfaces and programs at any time and instantly. On the other side, the Cloud Phone client could be implemented in any form, such as a cellphone, browser, or laptop. As a complete cloud-service-based phone solution, the Cloud Phone could provide the following benefits:

- It could make a low-cost smartphone possible, where the phone is just a terminal with basic communication functions and a screen decoder. Complicated applications become much simpler by shifting computing tasks to the cloud.
- It could emulate any existing devices (phones, game consoles, PDAs). It also could allow the user to enjoy the same personal experience on any phone.
- It could have complete knowledge of users' interactions in the cloud for potentially providing better services.

There still remains some future work required to deploy the services of Cloud Phone. For example, the offline experience should be taken into account, because the network might not always be available. Nevertheless, the concept of Cloud Phone inspires us to think about the next-generation mobile platforms with the consideration of cloud–mobile convergence. A middleware layer might exist on top of the phone OS, which could be more suitable than the general Web user interface for running rich cloud applications.

Conclusions

Cloud computing is not a simple extension of Web services. Breakthroughs may come with the introduction of a new application interface model between the cloud and the clients. User interaction with the cloud through client–cloud communications bounds the user experience and presents many technical challenges. We have shown that advanced multimedia compression and networking technologies can bridge the devices and the cloud effectively and efficiently, and potentially could help the evolution of cloud computing become a revolution. **MM**

References

1. M. Armbrust et al., ''A View of Cloud Computing,'' *Comm. ACM,* vol. 53, no. 4, 2010, pp. 50-58.
2. T. Richardson et al., ''Virtual Network Computing,'' *IEEE Internet Computing,* vol. 2, no. 1, 1998.
3. Microsoft, ''Remote Desktop Protocol,'' 2009; http://msdn.microsoft.com/en-us/library/aa383015.aspx.
4. H. Shen et al., ''A Proxy-Based Mobile Web Browser,'' *Proc. ACM Multimedia,* ACM Press, 2010, pp. 763-766.

Contact author Yan Lu at yanlu@microsoft.com.

Contact editor Wenjun Zeng at zengw@missouri.edu.