

Remote Display Solutions for Mobile Cloud Computing

Pieter Simoens, Filip De Turck, Bart Dhoedt,
and Piet Demeester, *Ghent University*



Proposed optimization techniques address the major challenges that varying wireless channel conditions, short battery lifetime, and interaction latency pose for the remote display of cloud applications on mobile devices.

Mobile devices have become an essential part of our daily life, with smartphone sales now surpassing desktop system sales.¹ As mobile device popularity grows, end-user demands to run heavier applications are also increasing.

Although advances in miniaturization continue, the desire to preserve the advantages mobile devices have over desktop systems in weight, size, and device autonomy will always impose intrinsic limits on processing power, storage capacity, battery lifetime, and display size. Researchers must redesign conventional desktop applications to operate on mobile hardware platforms, thereby often reducing functionality, whereas more demanding applications typically require specific hardware resources that are unlikely to be available on mobile devices.

At the same time, the Web hosts increasingly powerful computing resources, offering applications ranging from simple word processors to all-encompassing enterprise resource planning suites to 3D games.^{2,3} Both Microsoft and Google have developed complete online office suites—Office Live and Google Apps, respectively—that may emerge as all-round alternatives to their desktop office suites.

Cloud computing can broaden the range of applications available to mobile users beyond conventional office applications by supporting applications requiring graphical hardware, such as 3D virtual environments, or large storage capacity, such as 3D medical imaging. As multiple users share the cloud infrastructure, this technology can provide these hardware resources cost-effectively.

Mobile cloud computing offers a way to meet users' increasing functionality demands, as distant servers execute all application logic and only user interface functionalities reside on the mobile device. The device acts as a remote display, capturing user input and rendering display updates received from the server. However, short battery lifetime, varying wireless channel conditions, and interaction latency present major challenges for the remote display of cloud applications on mobile devices. Researchers have recently proposed several solutions to these issues.

MOBILE CLOUD COMPUTING CHALLENGES

Essentially, mobile cloud computing physically separates the user interface from the application logic. The mobile device executes only a viewer component, operating as a remote display for the applications running on distant servers in the cloud. As Figure 1 shows, a remote display framework has three components: a server-side component that intercepts, encodes, and transmits the application graphics to the client; a viewer component on the client; and a remote display protocol that transfers display updates and user events between both endpoints.

Using standard thin-client solutions such as Microsoft's Remote Desktop Protocol (RDP), Citrix Systems' Independent Computing Architecture (ICA), and AT&T's Virtual Network Computing (VNC) in a mobile cloud computing context is not straightforward. These architectures were originally designed for corporate environments, where

users connect over a wired local area network (LAN) to the central company server executing typical office applications. In this setting, the technical challenges are limited because delay and jitter are minimized, bandwidth availability is seldom a bottleneck, and office applications exhibit rather static displays compared to multimedia applications. In a mobile cloud computing environment, the remote display protocol must deliver complex multimedia graphics over wireless links and render these graphics on a resource-constrained mobile device.

A potential obstacle to mobile cloud computing success is the encumbered I/O functionality of mobile devices. Slideout keyboards and stylus devices facilitate user input and maximize display sizes without increasing overall device size, but they do not provide an adequate solution for convenient I/O. Novel user interfaces for mobile devices are, however, entering the market. Media tablets, such as Apple's iPad, offer a larger screen with touch functionality and a keyboard close in size to regular keyboards. Other manufacturers, such as NEC, minimize their device size and support external keyboards and displays⁴ to augment I/O functionality.

More fundamental challenges for mobile cloud computing lie in the short battery lifetime of mobile devices, the limited and varying bandwidth on wireless links, and the interaction latency between user input and display updates. Table 1 presents an overview of several solutions proposed to address these challenges.

DEVICE BATTERY LIFETIME

The operational time of mobile devices is often limited when they are extensively used. These battery capacity shortcomings result in short recharge cycles and prevent users from relying completely on their mobile device. During the past decade, advances in nominal battery capacity have been modest. Kostas Pentikousis⁵ observed that technological improvements are currently stagnating because of the lack of a major battery technology breakthrough comparable to the advent of rechargeable Li-ion batteries. Consequently, extending device auton-

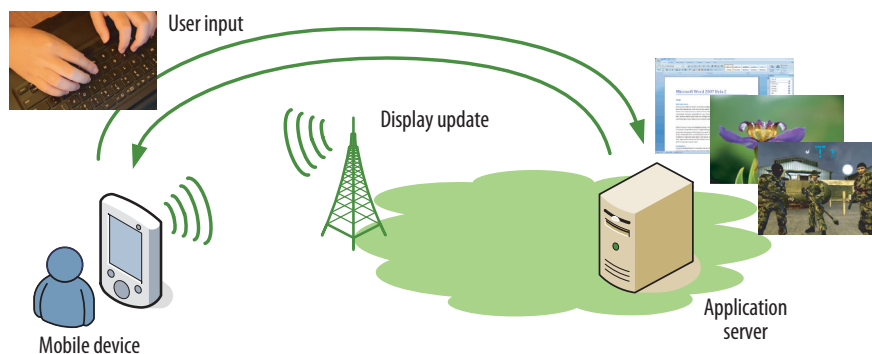


Figure 1. Remote display framework. The viewer component on the client forwards captured user input to the server. In turn, the server-side component intercepts, encodes, and transmits application output. A remote display protocol transfers display updates and user events between both endpoints.

Table 1. Proposed solutions to mobile cloud computing challenges.

Challenge	Solutions
Device battery lifetime	Cross-layer identification of wireless network interface card (WNIC) sleep intervals
Wireless bandwidth availability	Versatile graphics encoding Downstream data peak reduction Optimization of upstream packetization overhead
Interaction latency	Proximate cloudlet infrastructure Computing display updates in advance Image buffering for virtual environment streaming Scene object caching

omy should primarily be realized by making the device itself more energy efficient.

Offloading mobile applications

At first glance, offloading mobile applications to the cloud appears to be a straightforward way to save on device energy consumption because it reduces the amount of local processing. As Figure 2 shows, however, there is a tradeoff between local processing and network bandwidth consumption, and bidirectional communication with the application server incurs additional energy drain from the battery via the wireless network interface card (WNIC).

Karthik Kumar and Yung-Hsiang Lu⁶ modeled this tradeoff and concluded that, from an energy perspective, offloading applications from mobile devices is mainly advantageous when large amounts of computation are needed in combination with relatively small amounts of network communication. Demanding applications exchange a significant quantity of data between client and server because they entail a high degree of interactivity and detailed graphics—for example, walking around in a 3D virtual environment or rotating 3D medical images. Thus, according to Kumar and Lu's model, offloading such applications is not beneficial. On the other hand, demanding applications have important

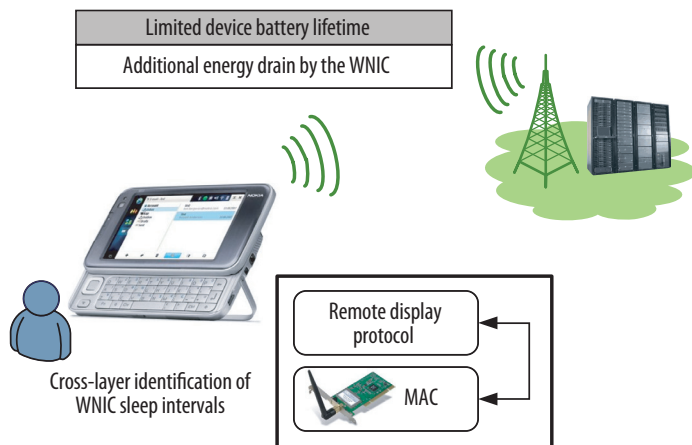


Figure 2. Offloading mobile applications to the cloud reduces the amount of local processing, but there is a tradeoff because bidirectional communication with the application server incurs additional drain from the battery via the wireless network interface card (WNIC). A proposed cross-layer power-saving approach operates between the media access control (MAC) layer and the remote display protocol layer.

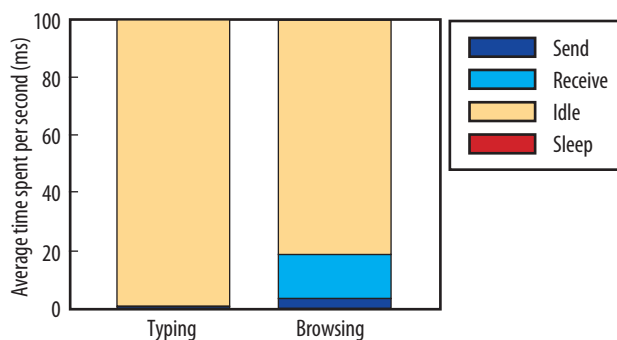


Figure 3. Average time distribution of the four possible WNIC modes—send, receive, idle, and sleep—resulting from remote display protocol traffic in two different scenarios: typing and browsing. The WNIC is mainly in idle mode and seldom enters sleep mode.

Table 2. Cisco Aironet WNIC power consumption.

Power mode	Measured power (mW)
Send	1,090-1,550
Receive	1,060-1,380
Idle	1,150
Sleep	300

hardware requirements that typically can only be met in the cloud and therefore need to be offloaded or remain unavailable to mobile users.

Cross-layer identification of WNIC sleep intervals

To develop strategies that optimize the energy balance, it is important to study WNIC energy consumption, which

is the product of the number of bytes exchanged over the wireless interface and the energy cost per byte. The average energy cost per byte is determined by the distribution of time over the four possible WNIC states: send, receive, idle, and sleep. Because a specific set of WNIC components are activated in each state, power consumption varies widely between the states. Figure 3 illustrates average WNIC time distribution in two typical remote display scenarios, typing and browsing, while Table 2 lists the power consumption in each state for a typical WNIC, the Cisco Aironet.⁷

Although the send and receive modes consume the most power, energy-saving approaches should focus on the large idle times observed in remote display scenarios. These idle times are a consequence of the limited frequency of user interactions imposed by the network round-trip time. After some interaction, users must wait until the results become visible on the screen before continuing their work. Furthermore, interactive applications will only update their display when instructed by the user—for example, by entering a URL or clicking on a hyperlink.

Major energy savings are expected when the WNIC transitions to energy-conserving sleep mode during these idle intervals. Sleep mode consumes three to five times less energy than idle mode because the radio interface is turned off. Of course, this implies that the WNIC will miss any incoming data when it is in sleep mode, so the sleep intervals must be carefully chosen.

A proposed cross-layer power-saving approach operates between the MAC layer and the remote display protocol layer (see Figure 2).⁸ Because the MAC layer operates on binary data and cannot discriminate between, for example, transmitted user input and Transmission Control Protocol (TCP) acknowledgments, it is unaware of the arrival of the next display update. The appropriate sleep intervals must therefore be determined at the remote display protocol layer, where the display update schedule is established—for example, via a push approach in which the server sends display updates with fixed intervals or a pull approach in which the client sends an explicit request. Correlating the transmission of user input to the network round-trip time predicts the arrival of the next display update. In between two display updates, the WNIC enters sleep mode. This sleep mode is interrupted at regular intervals to transmit user events. Researchers have used cross-layer optimization to reduce WNIC energy consumption by up to 52 percent.⁸

WIRELESS BANDWIDTH AVAILABILITY

Compared to fixed access networks, modern broadband mobile and wireless technologies offer limited and variable bandwidth availability. Universal Mobile Telecommunications System (UMTS) users typically receive up to 384 kilobits per second, while Krishna Balachandran and colleagues⁹ reported practical throughputs of 347 Kbps for Long Term Evolution (LTE) and up to 6.1 Mbps for WiMAX. Actual throughput depends on user mobility, interference, and fading effects.

Beyond the higher costs associated with limited broadband availability, other economic factors are motivating the development of highly efficient remote display compression technologies. Mobile users are increasingly confronted with volume-based subscription plans—such as that adopted by AT&T in 2010—and these users will not tolerate the sending of redundant bytes on the network. In fact, the lack of such technologies could impede development of new cloud applications.¹⁰

Versatile graphics encoding

As Figure 4 shows, the choice of codec to compress the intercepted application graphics at the server is a tradeoff between visual quality, compression efficiency, and decoding complexity.

Conventional remote display architectures including RDP, ICA, and VNC typically virtualize a layer of the graphic rendering stack at the server and forward intercepted drawing primitives to the client, such as instructions to draw a rectangle, display a bitmap, or put some text on the screen. This approach is optimal for applications, such as typical office applications, that only update small regions of the display or have a slow refresh rate with respect to the network round-trip time. Bandwidth requirements to remotely display this type of graphics do not exceed 200 Kbps and can be adequately served over wireless links.

On the other hand, encoding multimedia graphics applications would require numerous drawing primitives because they update large parts of the screen at high refresh rates and they often contain fine-grained and complex color patterns. This kind of graphics can be more efficiently encoded using a video codec, such as H.264 or MPEG-4. Using video codecs for remote display purposes is referred to as *interactive live streaming* because the graphics are mainly the result of user interaction, in contrast to regular video streaming, which requires only limited

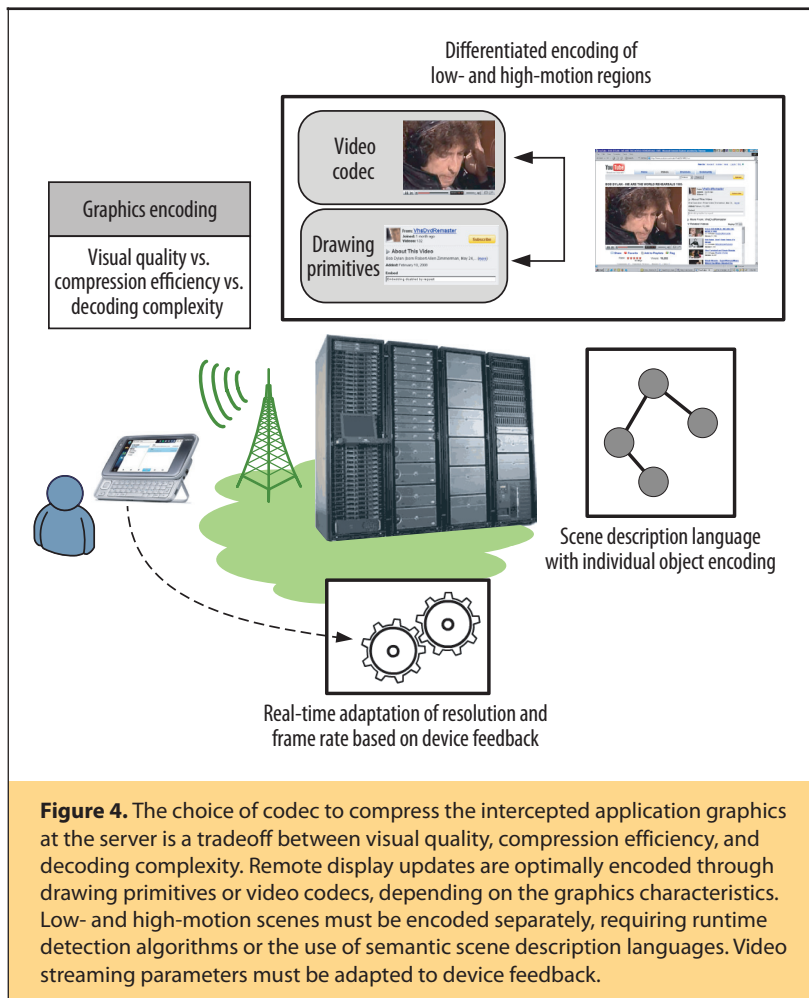


Figure 4. The choice of codec to compress the intercepted application graphics at the server is a tradeoff between visual quality, compression efficiency, and decoding complexity. Remote display updates are optimally encoded through drawing primitives or video codecs, depending on the graphics characteristics. Low- and high-motion scenes must be encoded separately, requiring runtime detection algorithms or the use of semantic scene description languages. Video streaming parameters must be adapted to device feedback.

user interaction—for example, to start and stop the video. Researchers have successfully applied interactive live streaming in remote 3D virtual environments² and games.¹¹

Even when only a single application is used, the characteristics of the graphics on the user display might significantly differ when a user is accessing mobile cloud computing services. For example, a user browsing a Wikipedia page might click on a link that opens a YouTube video in the same browser window. Remote display frameworks must therefore be able to switch seamlessly between multiple encoding modes based on an analysis of graphics at the server (see Figure 4).

Kheng-Joo Tan and colleagues¹² compared the pixels of subsequent frames and split each individual frame into low- and high-motion regions, respectively, encoding them through drawing commands or as H.264 video frames. This hybrid approach operates at the pixel level, which offers the advantage of cross-system applicability because it is the lowest layer of the rendering stack. Transparency at the pixel level, however, comes at the expense of losing any information on the nature of each object in the scene. Consequently, this technique uses the same encoding format

Table 3. Packetization overhead of TCP/IP headers when sending a single keystroke to the server.

Protocol	Payload (bytes)	Overhead (percent)
VNC Remote Framebuffer Protocol	8	83.33
Remote Desktop Protocol	6	86.96
Independent Computing Architecture	6	86.96

for low-motion regions regardless of whether they contain text characters or images.

Operating at a higher level in the rendering stack, Mihai Mitrea and colleagues¹³ intercepted high-level X11 drawing commands and encoded them using the MPEG-4 Binary Format for Scenes (BiFS), a powerful scene description language. Based on the intercepted X11 commands, they constructed an internal scene graph and converted it to BiFS semantics. A distinctive feature of BiFS is that this scene graph, containing the nature of each object in the scene, is binary encoded and streamed to the client. This allows encoding each object in its own optimal scheme.

While the characteristics of the application graphics mainly determine the choice of encoding format, the actual encoding parameters must be dynamically adapted to cope with wireless bandwidth fluctuations and heterogeneous mobile device capabilities. Numerous factors impact wireless link quality, including device antenna technology, distance from the access point, user speed and fading, and interference effects. At the same time, the various hardware configurations of commercial mobile devices induce variations in decoding power.

Gianluca Paravati and colleagues¹⁴ developed a closed-loop controller for interactive live streaming that optimizes the settings of the video codec parameters based on feedback from the client device. The mobile device regularly reports on the amount of data that the controller encodes per unit of time, a metric reflecting both the device hardware capabilities and the amount of data the device receives. By adjusting the resolution and image quality accordingly, the controller maintains a target frame rate that ensures a smooth visualization experience.

Downstream data peak reduction

Interactive applications only update their display when instructed by the user. These display updates usually involve sending a large amount of data to the client in a short interval, which requires an instantaneous bandwidth much higher than the average bandwidth requirement. Furthermore, this bursty traffic pattern is unfavorable in wireless network environments, as it might induce additional collisions on the wireless channel.

A study of this problem by Yang Sun and Teng-Tiow Tay¹⁵ revealed that repainting the graphical objects after

recurring user actions causes considerable redundancy in remote display protocol traffic. Their proposed hybrid cache-compression scheme uses the cached data as history to better compress recurrent screen updates, with the cache containing various drawing orders and bitmaps. Using RDP, they reduced the number of data spikes by 27 to 42 percent, depending on cache size, resulting in global network traffic reductions of 10 to 21 percent.

Optimization of upstream packetization overhead

User events are the principal source of upstream remote display traffic from client to server. Individually, each user event embodies only a small amount of information: a key or button ID, one bit to discriminate between the press and release actions, and possibly the current pointer coordinates. Nevertheless, user events induce important upstream traffic because they are often generated shortly after each other. Entering a single character results in two user events to indicate the press and release actions, whereas moving the mouse results in a sequence of pointer position updates.

Usually, the system transmits user events as they occur to minimize interaction latency. Because data packets sent upstream often contain a single user event, headers added at the TCP, IP, and wireless link layer cause large packetization overhead. Table 3 quantifies the packetization overhead of TCP/IP headers of three commonly used remote display protocols—the VNC Remote Framebuffer (RFB) Protocol, RDP, and ICA—when sending a single keystroke to the server. Optional headers and the wireless link layer header further increase total overhead.

Buffering user events at the client for a short period enables the joint transmission of multiple user events. The maximum buffering period is based on a tradeoff of remote display bandwidth reduction and interaction latency. A closed-loop controller running at the client integrates the interaction latency models in terms of this buffering period and the network round-trip time, ensuring that the average interaction latency does not exceed a predefined maximum value by continuously monitoring the current network status and adjusting the buffering period accordingly.¹⁶ The highest bandwidth reductions are achieved for interactive applications with frequent user events and lower round-trip times. For a text-editing scenario and network round-trip times less than 50 ms, researchers have achieved bandwidth reductions up to 78 percent.

INTERACTION LATENCY

While technological advances are likely to overcome bandwidth limitations, interaction latency—the delay users experience between generating some input and seeing the result on their display—is an intrinsic challenge of mobile cloud computing because the device must communicate even the most trivial user operations to the server.

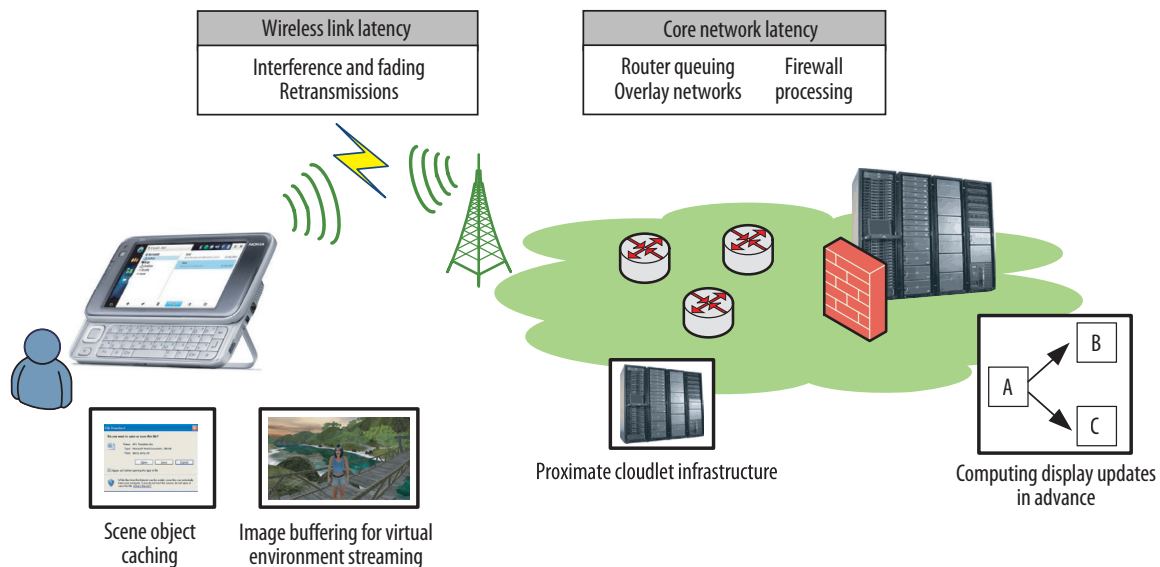


Figure 5. Strategies to mitigate interaction latency focus either on reducing the propagation delay by deploying the application on the proximate infrastructure or on reducing the synchronization between client and server by caching key objects or frames.

Niraj Tolia, David Andersen, and M. Satyanarayanan¹⁷ pointed out that these trivial interactions, such as moving the pointer to draw a line or to select some text, are far more problematic than loosely coupled tasks, such as Web browsing. The major difference is that users expect an immediate visual result from trivial operations, whereas they anticipate processing and download delays when clicking on a link.

Accustomed to the responsive interfaces of desktop applications, users will expect the same interactivity in a mobile cloud computing setting. As Figure 5 shows, remote display protocol data needs to traverse numerous links, both wireless and wired, and multiple network elements, each introducing additional propagation and transmission delays on the end-to-end path. Loss-correcting retransmissions on the wireless link, router queuing, suboptimal routing schemes, and firewall processing all entail significant propagation delays. Bandwidth limitations on the wireless link induce additional transmission delays, especially for immersive applications such as virtual environments that transfer highly detailed graphics from the server to the client. Sometimes several client-server interactions are required before a display update can appear on the screen—for example, when the server waits for the client's acknowledgment before sending the remainder of the data.

Solutions to mitigate interaction latency try to either reduce the number of hops on the end-to-end path by moving the application closer to the client or provide better synchronization mechanisms between client and server (see Figure 5).

Proximate cloudlet infrastructure

Satyanarayanan and colleagues¹⁸ introduced the concept of *cloudlets*: trusted, resource-rich computers dispersed over the Internet. Exploiting virtual machine technology, mobile devices rapidly deploy their services on the closest cloudlet by uploading an overlay VM to customize one of the generic base VMs commonly available on all cloudlets. The physical proximity ensures low-latency, one-hop, high-bandwidth wireless LAN access—for example, over the latest Wi-Fi 802.11n technology—instead of mobile radio technology access, such as high-speed downlink packet access (HSDPA) or LTE.

Despite its promise, the cloudlet concept may require the transfer of data from the central application server to nearby public infrastructure, which can be undesirable for security or privacy reasons. In these cases, latency optimization strategies must focus on reducing the number of round-trip times required to resynchronize the client device display with the server.

Computing display updates in advance

Given the current application state, the application server can predict potential display updates and stream them in advance to the client. Contrary to video streaming, in which the frame order is known in advance, in mobile cloud computing, the next display update depends on user input. For example, when a user opens an application menu, the server can precompute all dialog windows that can be opened by selecting one of the menu items.

Richard Pazzi, Azzedine Boukerche, and Tingxue Huang¹⁹ applied this approach to virtual 3D environ-

ments. Given the current user position, the system calculates the possible next user viewpoints in advance and provides them to the client. When the user actually moves forward, the client fetches the correct viewpoint from its cache.


Image buffering for virtual environment streaming

Due to limitations in mobile bandwidth and mobile device memory resources, in most cases, streaming all possible next display updates in advance is unfeasible. Furthermore, the gains of this precomputing technique are highly dependent on prediction accuracy. A better strategy might be to buffer some key display updates, for which the server only needs to provide a differential update. Boukerche, Pazzi, and Jing Feng²⁰ evaluated several cache management strategies and reduced the amount of requests during a 300-step movement in a 3D virtual environment from 300 to 145. Of course, in this case, the server response is still required to update the display.

Scene object caching

For more static applications, such as office applications, the potential next updates can be more accurately predicted as, for example, a menu layout will rarely change. Consequently, the number of corrective server updates will be more limited. A typical example would be the list of recently opened files in a text editor's File menu. Scene description languages such as MPEG-4 BiFS are particularly suited to support this client-side handling of user input.¹³ The client not only receives graphic updates, but also is informed about the structure of the displayed scene and its composing objects, as well as how the user can manipulate these objects.

By physically separating the user interface from the application logic, mobile cloud computing allows access to even the most demanding applications from intrinsically resource-constrained mobile devices. Developers tailor contemporary remote display optimization techniques to mobile devices' short battery lifetime, the varying and limited bandwidth availability on wireless links, and interaction latency. Although each of these solutions adequately addresses specific mobile cloud computing challenges, an overall approach is currently lacking.

Because of user mobility, the wide diversity of applications, and the varying wireless channel status, the mobile cloud computing context is highly dynamic. Future research should therefore focus on the design of a comprehensive framework that integrates the existing solutions and activates the most appropriate one depending on the current device, network, and cloud server status. 

References

1. C. Albanesius, "Smartphone Shipments Surpass PC Shipments for First Time. What's Next?" *PC Magazine*, 8 Feb. 2011; www.pcmag.com/article/2/0,2817,2379665,00.asp.
2. V.S. Pendyala and S.S.Y. Shim, "The Web as the Ubiquitous Computer," *Computer*, Sept. 2009, pp. 90-92.
3. F. Lamberti and A. Sanna, "A Streaming-Based Solution for Remote Visualization of 3D Graphics on Mobile Devices," *IEEE Trans. Visualization and Computer Graphics*, Mar./Apr. 2007, pp. 247-260.
4. H. Kawashima et al., "Virtual PC-Type Thin Client System," *NEC Technical J.*, Sept. 2007, pp. 42-47.
5. K. Pentikousis, "In Search of Energy-Efficient Mobile Networking," *IEEE Comm. Magazine*, Jan. 2010, pp. 95-103.
6. K. Kumar and Y.-H. Lu, "Cloud Computing for Mobile Users: Can Offloading Computation Save Energy?" *Computer*, Apr. 2010, pp. 51-56.
7. T. Kim et al., "An Energy-Aware Transmission Mechanism for WiFi-Based Mobile Devices Handling Upload TCP Traffic," *Int'l J. Comm. Systems*, May 2009, pp. 625-640.
8. P. Simoens et al., "Cross-Layer Optimization of Radio Sleep Intervals to Increase Thin Client Energy Efficiency," *IEEE Comm. Letters*, Dec. 2010, pp. 1095-1097.
9. K. Balachandran et al., "Performance Assessment of Next-Generation Wireless Mobile Systems," *Bell Labs Technical J.*, Feb. 2009, pp. 35-58.
10. C.C. Miller and B. Stone, "App Makers Worry as Data Plans Are Capped," *The New York Times*, 6 June 2010; www.nytimes.com/2010/06/07/technology/07data.html.
11. I. Nave et al., "Games@Large Graphics Streaming Architecture," *Proc. IEEE Int'l Symp. Consumer Electronics (ICSE 08)*, IEEE Press, 2008, pp. 205-208.
12. K.-J. Tan et al., "A Remote Thin Client System for Real-Time Multimedia Streaming over VNC," *Proc. IEEE Int'l Conf. Multimedia and Expo (ICME 10)*, IEEE Press, 2010, pp. 992-997.
13. M. Mitrea et al., "BiFS-Based Approaches to Remote Display for Mobile Thin Clients," *Proc. SPIE*, vol. 7444, 2009, p. 74440F, doi:10.1117/12.828152.
14. G. Paravati et al., "A Feedback-Based Control Technique for Interactive Live Streaming Systems to Mobile Devices," *IEEE Trans. Consumer Electronics*, Feb. 2010, pp. 190-197.
15. Y. Sun and T.T. Tay, "Analysis and Reduction of Data Spikes in Thin Client Computing," *J. Parallel and Distributed Computing*, Nov. 2008, pp. 1463-1472.
16. P. Simoens et al., "Upstream Bandwidth Optimization of Thin Client Protocols through Latency-Aware Adaptive User Event Buffering," *Int'l J. Communication Systems*, May 2011, pp. 666-690.

17. N. Tolia, D.G. Andersen, and M. Satyanarayanan, "Quantifying Interactive User Experience on Thin Clients," *Computer*, Mar. 2006, pp. 46-52.
18. M. Satyanarayanan et al., "The Case for VM-Based Cloudlets in Mobile Computing," *IEEE Pervasive Computing*, Oct.-Dec. 2009, pp. 14-23.
19. R.W.N. Pazzi, A. Boukerche, and T. Huang, "Implementation, Measurement, and Analysis of an Image-Based Virtual Environment Streaming Protocol for Wireless Mobile Devices," *IEEE Trans. Instrumentation and Measurement*, Sept. 2008, pp. 1894-1907.
20. A. Boukerche, R.W.N. Pazzi, and J. Feng, "An End-to-End Virtual Environment Streaming Technique for Thin Mobile Devices over Heterogeneous Networks," *Computer Comm.*, July 2008, pp. 2716-2725.

Pieter Simoens is a postdoctoral researcher in the Department of Information Technology at Ghent University, Belgium. His research focuses on the optimization of protocols to enable mobile cloud computing. Simoens received a PhD in electronic engineering from Ghent University. Contact him at pieter.simoens@intec.ugent.be.

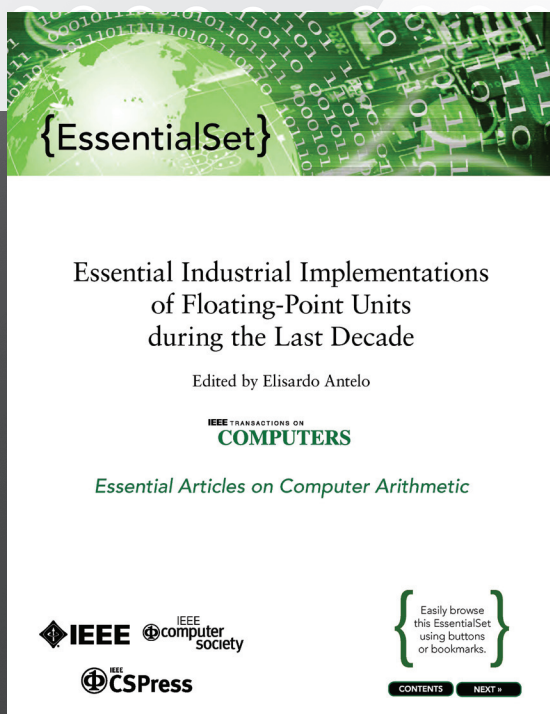
Filip De Turck is a part-time professor and a postdoctoral fellow in the Department of Information Technology at

Ghent University. His research interests include scalable software architectures for telecommunication networks and service management, and performance evaluation and design of new telecommunication services. De Turck received a PhD in electronic engineering from Ghent University. He is a member of IEEE and the ACM. Contact him at filip.deturck@intec.ugent.be.

Bart Dhoedt is a professor on the Faculty of Applied Sciences in the Department of Information Technology at Ghent University. His research interests include software engineering, distributed (autonomic) systems, grid and cloud computing, and thin-client computing. Dhoedt received a PhD in electronic engineering from Ghent University. He is a member of IEEE. Contact him at bart.dhoedt@intec.ugent.be.

Piet Demeester is a professor of communication networks at Ghent University, where he heads a research group active in the area of networks and distributed software and services (www.ibcn.intec.ugent.be). He received a PhD in electronic engineering from Ghent University. Demeester is an IEEE Fellow. Contact him at piet.demeester@intec.ugent.be.

cn Selected CS articles and columns are available for free at <http://ComputingNow.computer.org>.



NEW from  **CS Press**

ESSENTIAL INDUSTRIAL IMPLEMENTATIONS OF FLOATING-POINT UNITS DURING THE LAST DECADE: VOLUMES 1 & 2

Edited by Elisardo Antelo

Surveys the industrial design of floating-point units during the last decade. This EssentialSet is broken into two volumes, sold separately.

PDF edition • \$15 each (\$9 members) • 103 & 79 pp.

Order Online:
computer.org/store