Remote display solutions for mobile cloud computing

Pieter Simoens, Filip De Turck *member, IEEE*, Bart Dhoedt *member, IEEE*, Piet Demeester *Fellow, IEEE*.

***Abstract*—Although they suffer from intrinsic resource limitations, mobile devices have become very popular. Mobile cloud computing provides a solution to meet the increasing functionality demands of end-users, as all application logic is executed on distant servers and only user interface functionalities reside on the mobile device. The mobile device acts as a remote display, capturing user input and rendering the display updates received from the distant server. Varying wireless channel conditions, short battery lifetime and interaction latency introduce major challenges for the remote display of cloud applications on mobile devices. In this paper, we discuss a number of adequate solutions that have recently been proposed to tackle the main issues associated with the remote display of cloud services on mobile devices.**

I. Introduction

MOBILE devices have become an essential part of our daily life. Their portability is well appreciated by end-users and smartphones sales will soon surpass desktop sales. As mobile device popularity grows, end-user demands to run heavier applications are equally increasing. Although advances in miniaturization continue, the desire to preserve the advantages of weight, size and device autonomy will always impose intrinsic limits on processing power, storage capacity, battery lifetime and display size. Conventional desktop applications need to be redesigned to operate on mobile hardware platforms, thereby often losing functionality; whereas more demanding applications typically require specific hardware resources that are very unlikely to be available on mobile devices. At the same time, the web hosts increasingly powerful computing resources and has evolved to a ubiquitous computer, offering applications ranging from simple word processors, over all-encompassing enterprise resource planning suites to 3D games [1], [2]. Both Microsoft and Google, have developed complete online office suites, called Office Live and Google Apps respectively, that may evolve to all-round alternatives for the mobile office suites. Beyond the conventional office applications, cloud computing broadens the range of applications offered to mobile end-users with demanding applications in terms of graphical hardware, such as 3D virtual environments, or storage capacity, such as 3D medical imaging applications. As the cloud infrastructure is shared among multiple users, these hardware resources can be provided in a cost-effective way.

Essentially, the principle of mobile cloud computing physically separates the user interface from the application logic. Only a viewer component is executed on the mobile device, operating as a remote display for the applications running on distant servers in the cloud. Any remote display framework is composed of three components: a serverside 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. This is illustrated in Figure 1.

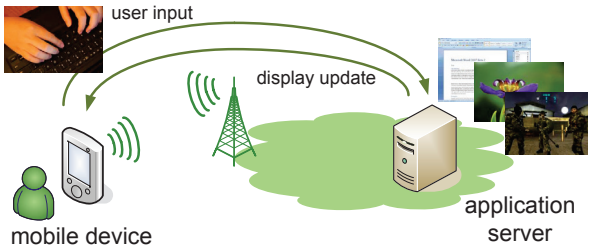

Fig. 1. The viewer component on the client forwards the captured user input to the server. In turn, the serverside component intercepts, encodes and transmits application output.

Using standard thin client solutions, such as Microsoft Remote Desktop Protocol (RDP), Citrix Independent Computing Architecture (ICA) and 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 to the central company server executing typical office applications. In this setting, the technical challenges are limited, because delay and jitter are small, bandwidth availability is seldom a limiting factor and office applications exhibit rather static displays when compared with multimedia applications. In a mobile cloud computing environment, the remote display protocol must be able to deliver complex multimedia graphics over wireless links and render these graphics on a resource constrained mobile device.

A potential blocker for the success of mobile cloud computing is the encumbered I/O functionality of mobile devices. Slideout keyboards and stylus handling are an attempt to facilitate user input and maximize display sizes without increasing the overall size of the device, but provide no adequate solution for convenient I/O. Media tablets, such as Apple's iPad, have recently become very popular. These slate devices have a larger screen with touch functionality on which a keyboard is presented of which the size comes close to regular keyboards. Other manufacturers, such as NEC, adhere to the principle keeping the size of their device minimal and support external keyboards and displays [3] to augment the I/O functionality.

More fundamental obstacles for mobile cloud computing

All authors are with the Dept. of Information Tech., Ghent University, Belgium e-mail: pieter.simoens@intec.ugent.be.

Digital Object Indentifier 10.1109/MC.2011.70 0018-9162/$26.00 2011 IEEE

emerge from the short battery lifetime of mobile devices, the limited and varying bandwidth on wireless links and the interaction latency between some user input and the update of the display. In the remainder of this article, we present and discuss a number of solutions that have recently been proposed to adequately address these challenges. An overview of the covered solutions is presented in Table I.

TABLE I
OVERVIEW OF RECENT SOLUTIONS TO THE CHALLENGES OF MOBILE CLOUD COMPUTING SURVEYED IN THIS ARTICLE.

challenge	solutions
battery lifetime (section II)	cross-layer identification of WNIC sleep opportunities [4]
wireless bandwidth availability (section III)	motion-based differentiated encoding [5] individual object encoding [6] real-time adaptation of encoding parameters [7]
interaction latency (section IV)	scene object caching [6] buffering of key images for virtual environments [8] proximate hosting infrastructure [9] computing display updates in advance [10]

II. OVERCOMING THE LIMITED MOBILE DEVICE BATTERY LIFETIME

The operational time of mobile devices is often limited when extensively used. These battery capacity shortcomings result in short recharge cycles and refrain users from relying completely on their mobile device. Over the last decade, the advances in nominal battery capacity have been modest. Pentikousis et al. [11] observe that the 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 autonomy should primarily be realized by making the device itself more energy efficient.

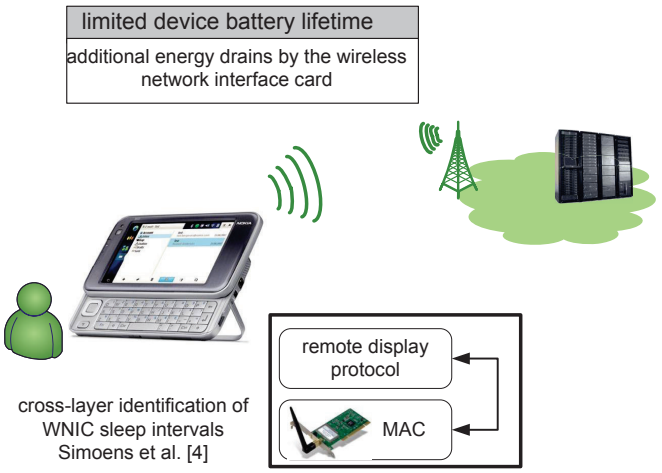

Fig. 2. The small form factor of mobile devices limits device autonomy and user interface functionalities.

At first sight, offloading applications to the cloud is a straightforward way to save on energy consumption because the amount of local processing is reduced. Local processing is however traded off with network bandwidth consumption, and the bidirectional communication with the application server incurs additional drains from the battery to the wireless network interface card (WNIC). Kumar et al. [12] have modeled this tradeoff and conclude that, from an energy perspective, offloading applications from mobile devices is mainly interesting when large amounts of computation are needed in combination with relatively small amounts of network communication. Demanding applications exchange a significant amount of data between client and server because they exhibit a high degree of interactivity and detailed graphics, e.g. walking around in a 3D virtual environment or rotating 3D medical images. According to Kumar's model, this is not beneficial from an energy perspective. On the other hand, as we have described above, these applications have important hardware requirements that are only available in the cloud. To optimize the energy balance, it is important to study the WNIC energy consumption and develop energy optimizing strategies.

The WNIC energy consumption is the product of the number of bytes exchanged over the wireless interface, and the energy cost per byte. Efficient compression techniques to reduce the amount of exchanged data are covered in section III of this article. The average energy cost per byte is determined by the distribution of the time over the four possible WNIC states: send, receive, idle and sleep mode. Because in each state a specific set of components is activated, the WNIC power consumption largely differs between the different states. Figure 3 visualizes our measurements on the average WNIC time distribution in typical remote display scenarios, while Table II contains figures on the typical power consumption in each WNIC state.

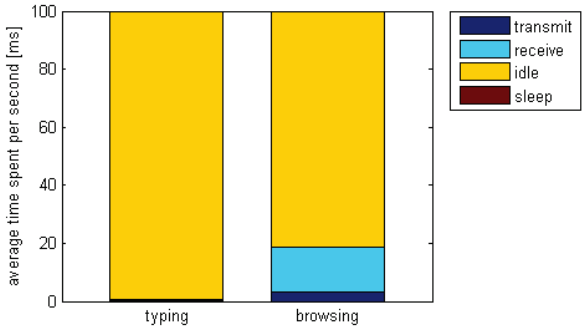

Fig. 3. Time distribution of the WNIC mode resulting from remote display protocol traffic in two different scenarios. The WNIC is mainly in idle mode and almost never enters sleep mode.

TABLE II
POWER CONSUMPTION OF THE CISCO AIRONET WNIC. TAKEN FROM KIM ET AL, *An energy-aware transmission mechanism for WiFi-based mobile devices handling upload TCP traffic* IN INT. J. COMMUN. SYST, 2009. THE VALUES ARE IN MW.

power mode	measured power (mW)
send	1090-1550
receive	1060-1380
idle	1150
sleep	300

Digital Object Indentifier 10.1109/MC.2011.70 0018-9162/$26.00 2011 IEEE

Although the transmit and receive mode are the most power consuming, power saving approaches should focus on the large idle times that are observed in remote display scenarios. These idle times are a consequence of the limited frequency of user interactions imposed by the network roundtrip time. After some interaction, users will 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, a browser display is only updated if the user enters a url or clicks on a hyperlink.

Potentially major energy savings are expected when the WNIC transitions to the energy conserving sleep mode during these idle intervals. The sleep mode is 3-5 times less energy consuming than the 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 and the sleep intervals must be carefully chosen. Simoens et al. [4] have developed a cross-layer power saving approach that operates between the MAC layer and the remote display protocol layer. Because the MAC layer operates on binary data and cannot discriminate between, for example, transmitted user input and TCP acknowledgements, it is unaware of the arrival of the next display update. The appropriate sleep intervals need to be determined at the remote display protocol layer where the display update schedule is established, e.g. a push approach in which the server sends display updates with fixed intervals or a pull approach where the client needs to send an explicit request. Simoens et al. correlate the transmission of user input to the network roundtrip time to predict the arrival of the next display update. In between two display updates, the WNIC is instructed to enter the sleep mode. These sleep modes are only interrupted at regular intervals to transmit user events. Through this cross-layer optimization, WNIC energy consumption reductions up to 52 % have been obtained.

III. Advanced display compression in a bandwidth limited wireless environment

Compared with fixed access networks, bandwidth availability on modern broadband mobile and wireless technologies is limited, variable and expensive. Typically, UMTS users receive up to 384 kbps, while Balachandran et al. [13] report practical throughputs of 347 kbps for LTE and up to 6.1 Mbps for WiMAX. Moreover, the actual throughput will vary due to user mobility and interference and fading effects. Besides technological limitations, economical considerations drive the demand for highly efficient remote display compression technologies. More and more, users are confronted with volume based subscription plans and hence will not tolerate any redundant byte to be sent on the network. For example, AT&T, a leading USA service provider, has adopted volume based pricing models in 2010. It is even expected that this economical dimension might impede the development of new cloud applications (www.nytimes.com/2010/06/07/technology/07data.html).

A. Versatile graphics encoding

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, such as Citrix ICA, AT&T VNC and Microsoft RDP virtualize the graphical library at the server and forward intercepted drawing primitives to the client, such as instructions to draw a rectangle, to display a bitmap or to put some text on the screen. This approach is optimal for applications that only update small regions of the display, or that have a slow refresh rate with respect to the network roundtrip time, such as typical office applications. Bandwidth requirements to remotely display this type of graphics do not exceed 200 kbps, and can be perfectly served over wireless links. On the other hand, a lot of drawing primitives would be required to encode the graphics of multimedia applications, because these update large parts of their screen at high refresh rates and their graphics often contain fine-grained and complex color patterns. This kind of graphics can be more efficiently encoded by means of a video codec, such as H.264 or MPEG-4 video. 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 with only limited user interaction, e.g. to start and stop the video. Interactive live streaming has been applied successfully in the context of remote 3D virtual environments [2] and gaming [14].

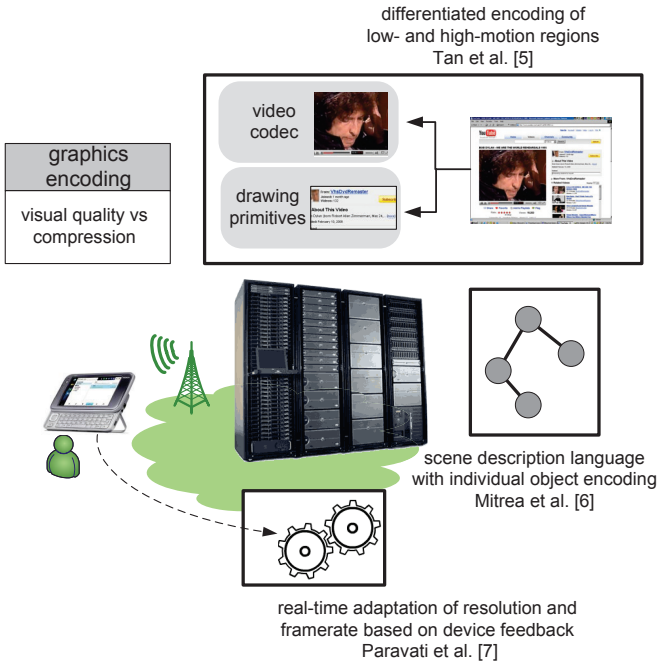

Fig. 4. Display updates are optimally encoded through drawing primitives or video codecs, dependent on the characteristics of the graphics. Low- and high-motion scenes must be separately encoded, requiring run-time detection algorithms or the usage of semantic scene description languages. Video streaming parameters need to be adapted to device feedback.

Even when only a single application is used, the characteristics of the presented graphics on the user display may largely differ during the period a user is accessing the mobile cloud computing services. For example, users may browse to a Wikipedia page and subsequently click on link that opens a YouTube video in the same browser window. Remote display frameworks must therefore be able to switch

Digital Object Indentifier 10.1109/MC.2011.70 0018-9162/$26.00 2011 IEEE

seamlessly between multiple encoding modes, based on an analysis of graphics at the server. Tan et al. [5] compare the pixels of subsequent frames to split each individual frame in low- and high-motion regions, which are respectively encoded 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. The transparency at the pixel level comes at the expense of losing any information on the nature of each object in the scene. Consequently, the same encoding format will be used for low-motion regions irregardless if they contain text characters or images. Mitrea et al. [6] operate at a higher level in the rendering stack. They intercept high-level X11 drawing commands and encode it through the MPEG-4 Binary Format for Scenes (BiFS), which is a powerful scene description language. Based on the intercepted X11 commands, an internal scene graph is constructed and converted 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 to encode each object in its own optimal encoding scheme.

While the choice of encoding format is mainly determined by the characteristics of the application graphics, the actual encoding parameters need to be dynamically adapted to cope with wireless bandwidth fluctuations and heterogeneous mobile device capabilities. Numerous factors impact wireless link quality, such as 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. Paravati et al. [7] have 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 is encoded per unit of time, a metric reflecting both the device hardware capabilities, as well as of the amount of data that is received by the device. By adjusting the resolution and image quality accordingly, the controller aims to maintain a target frame rate to ensure a smooth visualization experience.

B. Downstream data peak reduction

Interactive applications only update their display unless instructed by the user. Usually, these display updates involve a large amount of data that needs to be sent to the client in a short interval to swiftly update the display. The delivery of these data burst requires an instantaneous bandwidth that is much higher than the average bandwidth requirement. Furthermore, this bursty traffic pattern is unfavourable in wireless network environments, as it might induce additional collisions on the wireless channel. Sun et al. [15] have studied this problem. Their analysis of remote display protocol traffic traces reveals a lot of redundancy, caused by the repainting of graphical objects after recurring user actions. They propose a hybrid cache-compression scheme whereby the cached data is used as history to better compress recurrent screen updates. The cache contains various drawing orders and bitmaps. Using Microsoft's Remote Display Protocol (RDP) and dependent on the size of the cache, they are able to reduce the number of data spikes by 27-42 %, which results in global network traffic reductions of 10-21 %.

C. Optimization of upstream packetization overhead

User events are the principal source of remote display traffic in the upstream direction 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 action 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 action, whereas moving the mouse results in a sequence of pointer position updates. Usually, user events are transmitted as they occur to minimize interaction latency. Because data packets sent upstream often contain a single user event, a large packetization overhead is observed owing to the headers added at the TCP, IP and (wireless) link layer. Table III quantifies the upstream packetization overhead of three commonly used remote display protocols.

TABLE III
PACKETIZATION OVERHEAD OF TCP/IP HEADERS WHEN SENDING A SINGLE KEY STROKE TO THE SERVER. THE TOTAL OVERHEAD IS FURTHER INCREASED BY OPTIONAL HEADERS AND THE WIRELESS LINK LAYER HEADER.

protocol	payload [bytes]	overhead [%]
VNC RFB	8	83.33
Microsoft RDP	6	86.96
Citrix ICA	6	86.96

By buffering user events at the client for a short period, multiple user events can be jointly transmitted. The maximum buffering period is a consideration of remote display bandwidth reduction against interaction latency. Simoens et al. [16] developed models of the interaction latency in terms of this buffering period and the network roundtrip time. These models are integrated in a closed-loop controller running at the client, which ensures 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. The highest bandwidth reductions are achieved for interactive applications with frequent user events and lower roundtrip times. For a text editing scenario and network roundtrip times below 50 ms, Simoens et al. achieve bandwidth reductions up to 78 %.

IV. Ensuring crisp interaction response

Interaction latency, i.e. the delay a user experiences between generating some user input and having the result presented on his display, is key challenge of mobile cloud computing. Whereas bandwidth limitations are likely to disappear with technological advancements, interaction latency is an intrinsic key challenge of mobile cloud computing because even the most trivial user operations need to be communicated to the server. Tolia et al. [17] point out that these trivial interactions,

Digital Object Indentifier 10.1109/MC.2011.70 0018-9162/$26.00 2011 IEEE

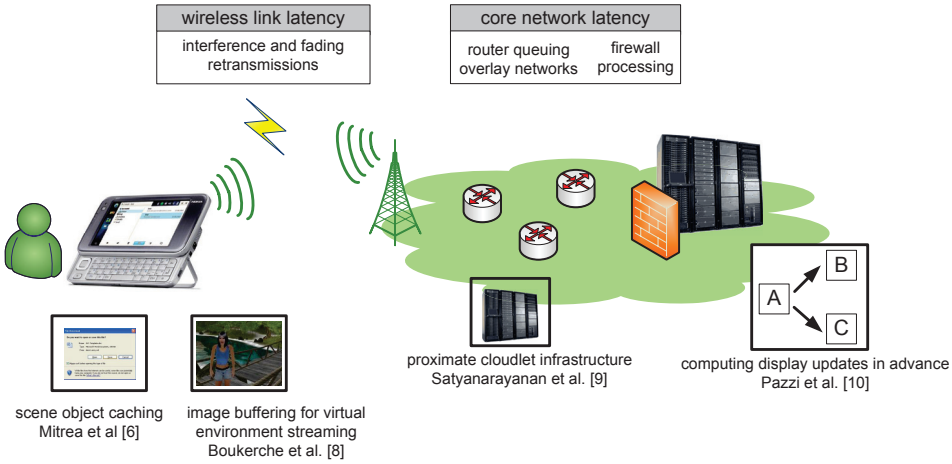

Fig. 5. Strategies to mitigate the interaction latency either focus on reducing the propagation delay by deploying the application on proximate infrastructure, or on reducing the synchronization between client and server by caching key objects or frames.

such as moving the pointer to draw a line or to select some text, are far more challenging than loosely coupled tasks, such as web browsing. The major difference is that users expect an immediate visual result of 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. Remote display protocol data needs to traverse numerous links, both wireless and wired, and numerous 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 entail important 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 to the client. Sometimes several client-server interactions are required before a display update can be shown on the screen, e.g. when the server waits for the acknowledgement of the client before sending the remainder of the data.

Solutions to mitigate the interaction latency either target a reduction of the number of hops on the end-to-end path by moving the application closer to the client, or better synchronization mechanisms between client and server. Satyanarayanan et al. [9] introduce the concept of *cloudlets*: trusted, resource-rich computers that are dispersed over the Internet. Exploiting virtual machine technology, mobile devices rapidly deploy their services on the most nearby cloudlet by uploading an overlay virtual machine to customize one of the generic base virtual machines that are commonly available on all cloudlets. The physical proximity ensures low-latency, one-hop, high-bandwidth wireless LAN access, e.g. over the latest WiFi 802.11n technology, instead of mobile radio technology access, such as HSDPA or LTE.

Although the cloudlet concept is very promising, it may require the transfer of data from the central application server to nearby public infrastructure. This can be undesirable for security or privacy reasons. In these cases, latency optimization strategies need to focus on a reduction of the number of roundtrip times that is required to resynchronize the client device display with the server. Given the current application state, the application server can predict potential display updates and stream these in advance to the client. Contrary to video streaming, where 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 could precompute all dialog windows that can be opened by selecting one of the menu items. Pazzi et al. [10] have applied this precomputing technique to the use case of virtual 3D environments. Given the current user position, the possible next user viewpoints are calculated in advance and provided to the client. When the user actually moves forward, the client fetches the correct viewpoint from its cache.

Due to the limitations in mobile bandwidth and mobile device memory resources, it is in most cases unfeasible to stream in advance all possible next display updates. Furthermore, the gains of this precomputing technique highly depend on the prediction accuracy. A better strategy might be to buffer only a number of key display updates, for which the server only needs to provide a differential update. Boukerche et al. [8] have evaluated a number of cache management strategies and are able to reduce 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. For more static applications, e.g. office applications, the potential next updates can be more accurately predicted as, for example, the layout of a menu will almost never change. Consequently, the number of corrective server updates will be more limited. One typical example would be the list of recently opened files in the File menu of a text editor. Scene description languages such as MPEG-4 BiFS are particularly suited to support this clientside handling of user input. The client not only receives graphic updates, but is also informed on the structure of the displayed scene and its composing objects, as well as on how the user can manipulate these objects.

Digital Object Indentifier 10.1109/MC.2011.70 0018-9162/$26.00 2011 IEEE

V. Conclusion

By physically separating the user interface from the application logic, the principle of mobile cloud computing allows to access even the most demanding applications in the cloud from intrinsically resource-constrained mobile devices. In this article, we have surveyed contemporary remote display optimization techniques specifically tailored to the short mobile device battery lifetime, the varying and limited bandwidth availability on wireless links and the interaction latency. Although each of these solutions adequately address specific challenges of mobile cloud computing, an overall approach is currently lacking. The context of mobile cloud computing is highly dynamic, owing to the user mobility, the wide diversity of applications, and the varying wireless channel status. Future research should therefore be devoted to the design of an overall framework, integrating all the presented solutions, and activating the most appropriate solutions dependent on the current device, network and cloud server status.

References

[1] V. S. Pendyala and S. S. Y. Shim, "The Web as the Ubiquitous Computer," *COMPUTER*, vol. 42, no. 9, pp. 90–92, SEP 2009.

[2] F. Lamberti and A. Sanna, "A streaming-based solution for remote visualization of 3D graphics on mobile devices," *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS*, vol. 13, no. 2, pp. 247–260, MAR-APR 2007.

[3] H. Kawashima, K. Koshiba, K. Tuchimochi, K. Futamura, M. Enomoto, and M. Watanabe, "Virtual PC-type thin client system," *NEC TECHNICAL JOURNAL*, vol. 2, no. 3, pp. 42–47, SEP 2007.

[4] P. Simoens, F. A. Ali, B. Vankeirsbilck, L. Deboosere, F. De Turck, B. Dhoedt, P. Demeester, and R. Torrea-Duran, "Cross-Layer Optimization of Radio Sleep Intervals to Increase Thin Client Energy Efficiency," *IEEE COMMUNICATIONS LETTERS*, vol. 14, no. 12, pp. 1095–1097, DEC 2010.

[5] K.-J. Tan, J.-W. Gong, B.-T. Wu, D.-C. Chang, H.-Y. Li, Y.-M. Hsiao, Y.-C. Chen, S.-W. Lo, Y.-S. Chu, and J.-I. Guo, "A remote thin client system for real time multimedia streaming over VNC," in *2010 IEEE International Conference on Multimedia and Expo (ICME)*, 2010 2010, pp. 992–7.

[6] M. Mitrea, P. Simoens, B. Joveski, J. Marshall, A. Taguengayte, F. Preteux, and B. Dhoedt, "BiFS-based approaches to remote display for mobile thin clients," in *Proceedings of the SPIE - The International Society for Optical Engineering*, vol. 7444, 2009 2009, p. 74440F (8 pp.).

[7] G. Paravati, C. Celozzi, A. Sanna, and F. Lamberti, "A Feedback-Based Control Technique for Interactive Live Streaming Systems to Mobile Devices," *IEEE TRANSACTIONS ON CONSUMER ELECTRONICS*, vol. 56, no. 1, pp. 190–197, FEB 2010.

[8] 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 COMMUNICATIONS*, vol. 31, no. 11, pp. 2716–2725, JUL 15 2008.

[9] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing," *IEEE PERVASIVE COMPUTING*, vol. 8, no. 4, pp. 14–23, OCT-DEC 2009.

[10] 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 TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT*, vol. 57, no. 9, pp. 1894–1907, SEP 2008.

[11] K. Pentikousis, "In Search of Energy-Efficient Mobile Networking," *IEEE COMMUNICATIONS MAGAZINE*, vol. 48, no. 1, pp. 95–103, JAN 2010.

[12] K. Kumar and Y.-H. Lu, "CLOUD COMPUTING FOR MOBILE USERS: CAN OFFLOADING COMPUTATION SAVE ENERGY?" *COMPUTER*, vol. 43, no. 4, pp. 51–56, APR 10 2010.

[13] K. Balachandran, Q. Bi, A. Rudrapatna, J. Seymour, R. Soni, and A. Weber, "Performance Assessment of Next-Generation Wireless Mobile Systems," *BELL LABS TECHNICAL JOURNAL*, vol. 13, no. 4, pp. 35–58, WIN 2009.

[14] I. Nave, H. David, A. Shani, Y. Tzruya, A. Laikari, P. Eisert, and P. Fechteler, "Games@Large graphics streaming architecture," in *2008 IEEE INTERNATIONAL SYMPOSIUM ON CONSUMER ELECTRONICS, VOLS 1 AND 2*, ser. IEEE International Symposium on Consumer Electronics, 2008, pp. 205–208, IEEE International Symposium on Consumer Electronics, Vilamoura, PORTUGAL, APR 14-16, 2008.

[15] Y. Sun and T. T. Tay, "Analysis and reduction of data spikes in thin client computing," *JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING*, vol. 68, no. 11, pp. 1463–1472, NOV 2008.

[16] P. Simoens, B. Vankeirsbilck, L. Deboosere, F. A. Ali, F. De Turck, B. Dhoedt, and P. Demeester, "Upstream bandwidth optimization of thin client protocols through latency-aware adaptive user event buffering," *Int. J. Commun. Syst.*, Accepted for publication [available online], 2010.

[17] N. Tolia, D. Andersen, and M. Satyanarayanan, "Quantifying interactive user experience on thin clients," *COMPUTER*, vol. 39, no. 3, pp. 46+, MAR 2006.

Digital Object Indentifier 10.1109/MC.2011.70 0018-9162/$26.00 2011 IEEE