Performance Evaluation of Remote Display Access for Mobile Cloud Computing

Youming Lin^{a,b}, Teemu Kämäräinen^a, Mario Di Francesco^a, Antti Ylä-Jääski^a

^a*Department of Computer Science, Aalto University School of Science*

^b*Centre for Strategic Infocomm Technologies, Singapore*

Abstract

Mobile applications are increasingly exploiting cloud computing to overcome the resource limitations of mobile devices. To this end, the most computationally expensive tasks are offloaded to the cloud and the mobile application simply interacts with a remote service through a network connection. One way to establish such a connection is given by remote display access, in which a mobile device just operates as a thin client by relaying the input events to a server and updating the screen based on the content received. In this article, we specifically address remote access as a means for mobile cloud computing, with focus on its power consumption at mobile devices. Different from most of the existing literature, we take an experimental approach based on real user sessions employing different remote access protocols and types of applications, including gaming. Through several experiments, we characterize the impact of the different protocols and their features on the power consumption and the network utilization. We conclude our analysis with considerations on usability and user experience.

Keywords: Experimental evaluation, mobile cloud computing, energy-efficient networking, remote access protocols, cloud gaming.

Email addresses: lin.ym@csit.gov.sg (Youming Lin), teemu.kamarainen@aalto.fi (Teemu Kämäräinen), mario.di.francesco@aalto.fi (Mario Di Francesco), antti.yla-jaaski@aalto.fi (Antti Ylä-Jääski)

Preprint submitted to Elsevier

March 16, 2015

1. Introduction

Mobile devices such as smartphones and tablets have become very popular, and an increasing amount of people are nowadays using them not only for entertainment but also for several other activities. However, despite the advances in hardware manufacturing and in wireless communication technologies, mobile devices are still much more resource-constrained than desktop computers and workstations. The fundamental limitation of smartphones and tablets is represented by their battery, which provides a limited amount of energy before it needs to be recharged, thus significantly limiting utilization on the move [1].

In order to overcome the resource constraints of mobile devices, several solutions have been proposed. One of the most important is represented by *offloading* computational and storage resources to the cloud [2, 3]. With this approach, the mobile device runs only part of an application, namely, the components which are less resource-intensive. The rest of the application is deployed in the cloud and is accessed through a wireless network connection. One of the major challenges in offloading applications to the cloud is represented by the overhead in coordinating the different components and accessing the data of interest. Such an overhead may be significant, not only as network traffic, but also in terms of power consumption [4, 5].

A different option is given by *remote display access* [6]. With this approach, mobile devices use a thin software client and connect to a remote display server running an operating system as well as one or more applications. The thin client displays the graphical user interface of the remote applications on the mobile device and relays all input events to the server. The server then processes the inputs and sends back to the client the changes in the content to be displayed. When the server is virtualized and runs in a data center, remote display access actually corresponds to an extreme case of *mobile cloud computing*, where the mobile device only operates as an input/output device, while most of the computations are handled by the remote system [7, 8].

In contrast with approaches specifically targeted to mobile cloud computing,

most of the solutions for remote display access were originally designed for the reference scenario represented by personal computers. In that context, clients are static, with a wired connection to the Internet, and have plenty of resources. Some solutions have been specifically designed for remote access with mobile devices [9, 10]; however, they are not usually available to the end users, or they cannot easily be integrated into the existing infrastructure. As a consequence, the vast majority of remote access protocols available for mobile devices are still those designed for personal computers. Even though some application-specific solutions such as [11] were proposed, they have not been evaluated as general means for mobile cloud computing.

In this article, we characterize the power consumption of mobile cloud computing realized through remote display technologies. Specifically, we analyze the performance of widely-used remote access protocols, including Virtual Network Connection (VNC), Remote Desktop Protocol (RDP) and GamingAnywhere. We perform several experiments involving realistic usage scenarios under different wireless communication technologies, i.e., WiFi and Long Term Evolution (LTE). Moreover, we characterize the power consumption and bandwidth utilization of the different solutions considered and relate them to the protocol features. Finally, we provide some considerations on aspects related to usability and user experience.

The remainder of the article is organized as follows. Section 2 discusses the related work, while Section 3 introduces the relevant background about remote access protocols and energy profiling of mobile devices. Section 4 presents the experimental setup and Section 5 discusses the obtained results. Finally, Section 6 concludes the article.

2. Related work

In the following, we discuss the most relevant solutions proposed in the existing literature in the context of virtual networking and mobile computing. In detail, we first overview the major results about performance evaluation of remote access protocols, then focus on solutions specifically targeted to cloud

environments and mobile devices. For clarity, we group the related work in different subsections accordingly.

2.1. Performance of Remote Display Protocols

The performance of different remote display protocols was evaluated in [6, 12, 13]. Specifically, the authors performed several experiments on a real testbed under varying workloads corresponding to web-browsing and multimedia-oriented user activities. To this end, the evaluation exploited a *slow-motion* benchmarking technique, i.e., modified application benchmarks in which the activities affecting display updates are separated by adding artificial delays. Slow-motion benchmarking allows to accurately measure the impact between a single user-generated event at the client and its counterpart at the server. However, such an approach alters the actual interactions, possibly resulting in unexpected side-effects. Furthermore, it is based on synthetic application benchmarks rather than on the actual input from real users. In contrast, in this article we explicitly consider sessions by real users instead of relying on artificial benchmarks.

A performance study of user experience with thin clients was carried out in [14]. The authors recorded real remote display sessions using the Virtual Network Computing (VNC) protocol and involving interactive programs for image as well as document creation. The related traces were then used to perform extensive simulations in a wide range of network scenarios, characterized by different values of bandwidth and delay. As the work focused on user experience, the evaluation was limited to classify the user satisfaction based on the response time. The work in this article shares some similarities with [14], as we explicitly consider real use cases and adopt a record-replay strategy in our evaluation. In contrast, we perform experiments on a real testbed and specifically consider quantitative metrics (i.e., power consumption and network utilization) in our evaluation.

Finally, none of the solutions mentioned above considered scenarios involving virtualized servers nor mobile clients. Instead, we specifically evaluate remote display access as a means for mobile cloud computing. To this end, we perform

measurements on a real testbed consisting of a mobile device and a virtualized server instance.

2.2. Cloud-based Solutions

Offloading resources to the cloud has been the major focus for improving the performance and reducing the power consumption of mobile devices [4]. Some approaches have focused on techniques for automated and even dynamic offloading of mobile applications to the cloud. Among them, MAUI [2] is a platform which can evaluate the energy-cost tradeoff of applications running on top of the Microsoft .NET framework. Based on such an evaluation, MAUI automatically decides to execute part of an application in the cloud rather than on the mobile device. ThinkAir [15] uses a similar approach but targets Android devices and provides a finer granularity for offloading decisions. Different from those solutions, CloneCloud [3] enables offloading Android applications to the cloud without the need of special support by the application developer, who would otherwise need to annotate the application code to exploit the offloading platform. Due to the focus on computational and storage resources, the solutions for offloading mentioned above address computing aspects more than networking ones. As a consequence, they have been evaluated in use cases that are not very practical. In contrast, we consider realistic use cases and specifically address remote display scenarios.

Some research on offloading proposed running native mobile applications in the cloud and accessing them through a virtual network connection. Among them, the work in [16] evaluated PDF viewing and web browsing performed locally on a phone and remotely on a server running the X11 protocol over WiFi. The concept of *virtual smartphone* was introduced in [17], which also presented an architecture suitable to support the remote installation and execution of mobile applications. The approach in [17] relies on the VNC protocol for remote access, but it does not provide any characterization of the related resource utilization. In this article, instead, we characterize the power consumption and the bandwidth utilization of several remote display access protocols, including

(but not limited to) VNC.

The concept of *virtualized screen* was introduced in [8], together with an architecture for thin-client mobile devices to perform screen rendering in the cloud. The proposed model targets complex graphic environments based on 3D rendering and takes a cross-layer approach which is both network and content-aware. Even though the architecture of the proposed system is described in detail, the solution in [8] is not evaluated in real-world scenarios. In contrast, we experimentally evaluate the power consumption of different remote display solutions available off-the-shelf under realistic use cases.

A cloud-based architecture for providing desktop services to thin clients was proposed in [18]. The authors outline the requirements for a smooth user experience at the clients and specifically consider issues related to scalability, resource allocation, and energy costs in a data center. The work takes a provider-centric perspective; it does not evaluate different remote desktop solutions and their impact on the power consumption of mobile devices. Indeed, we evaluate the performance of remote desktop access as a means for mobile cloud computing.

2.3. Mobile-friendly Remote Access

Solutions for remote display access have originally targeted personal computers [19]. Approaches suitable for mobile devices have been designed only recently. For instance, MobiDesk was proposed in [9] as an infrastructure for mobile virtual desktop computing. MobiDesk uses different techniques to provide display and network virtualization for both local and wide area networks, including session migration. However, this solution mainly targets mobile laptops and not devices such as smartphones or tablets which have significantly lower energy resources. In contrast, we specifically consider energy-constrained mobile devices in this article.

SmartVNC [10] is a solution built on top of VNC to improve the user experience in remote computing scenarios. The proposed approach operates directly at the user interface level and records the sequence of operations performed by the user as application-independent *smart-macros*. These macros then trigger the

1
2
3
4
5
6
7
8
9 playback of procedural operations with a single tap or gesture, thus eliminating
10 cumbersome or redundant operations involving the user interface at the mobile
11 device. The goal of SmartVNC is to increase the productivity and efficiency of
12 an existing remote desktop protocol, with focus on the user experience. Differ-
13 ent from that solution, we specifically address the power consumption of remote
14 display access at the mobile device by considering several existing protocols,
15 including (but not limited to) VNC.
16
17
18
19

20 Remote display solutions for mobile cloud computing were surveyed in [7].
21 The authors recognized battery lifetime as one of the major challenges for cloud
22 access from mobile devices and highlighted the importance of offloading and
23 cross-layer design as key aspects of energy-efficient cloud computing. However,
24 they did not provide any quantitative evaluation of the power consumption,
25 while in this article we present results obtained through measurements in real-
26 istic scenarios.
27
28
29
30

31 A high-level approach for delivering multimedia data to mobile devices was
32 proposed in [20]. The proposed solution takes a high-level approach based on
33 a multimedia scene graph, i.e., a hierarchical representation of multimedia ob-
34 jects providing abstractions of their properties. Specifically, the work builds on
35 related features of the MPEG-4 standard and proposes a remote display system
36 for mobile thin clients. Simulation experiments showed that the proposed solu-
37 tion uses significantly less bandwidth than VNC and Remote Desktop Protocol
38 (RDP). However, the evaluation is limited to a mobile client connected to a local
39 server through WiFi. In this article, instead, we specifically consider a display
40 server deployed in the cloud. Furthermore, we use not only WiFi but also Long
41 Term Evolution (LTE) as the communication technologies at the mobile client.
42
43
44
45
46
47

48 Several solutions have been developed for the specific scenario represented by
49 cloud gaming. For instance, GamingAnywhere [21, 11] is an open-source cloud
50 platform that allows users to play a game in the cloud and receive the displayed
51 content as a video stream. There are a few studies about the performance of
52 remote gaming, such as [22, 23, 24], but they mostly evaluate aspects related to
53 user-experience (e.g., response time). In contrast, we specifically focus on the
54
55
56
57
58
59
60
61
62
63
64
65

power consumption and the network utilization of remote display access in the context of mobile cloud computing.

Modeap [25] is a web-based cloud service to run desktop applications on mobile devices. The proposed approach operates at the graphical user interface of the desktop server and translates different classes of elements (such as text, images, and basic drawing primitives) into sequences of commands that are replayed by the mobile client through a web browser. Despite specifically targeting cloud scenarios, Modeap does not explicitly consider the impact of virtualization on the performance, neither characterizes the power consumption at the mobile device. In this work, instead, we address both the above-mentioned aspects.

Preliminary results of this work have appeared in [26] where a local connection between a mobile device and a regular workstation was considered. In this article, instead, we employ a remote display server running in a virtualized instance at a cloud provider. Furthermore, we have entirely redesigned the experimental methodology and the considered scenarios by providing a comprehensive evaluation for different wireless communication technologies.

3. Background

In this section, we briefly overview the background aspects that will be useful in the rest of the discussion. Before proceeding further, let us introduce the reference architecture, which is illustrated in Figure 1. On one side, there is a mobile device running a remote display client. On the other side, there is a cloud-based data center running a remote display server within a virtualized instance. Client and server are connected through an IP-based network, and the mobile device uses a wireless communication technology to access the remote display. Both ends agree on a specific remote access protocol to support the remote display session.

3.1. Remote Access Protocols

There are several remote display solutions available, but many of the leading products are based on a few technologies and protocols. Among them, we

Figure 1: Reference architecture for the considered remote display access scenario.

consider Virtual Network Computing (VNC) and Remote Desktop Protocol (RDP), as they are the most widespread solutions and they are also available for a wide range of operating systems as well as mobile device platforms. We also consider the GamingAnywhere system as it is general enough to support different scenarios, even though it has been designed with reference to a specific application (i.e., cloud gaming).

VNC [19] is an established multi-platform remote desktop solution. It is built on top of the Remote Framebuffer (RFB) protocol, which evolved over time to include extensions and optimizations introduced by software derived from the original VNC server [27]. RFB, as the name suggests, operates at the framebuffer level and is based on primitives which can draw rectangular regions on the screen. Each region can be encoded in different ways and a sequence of drawing primitives produces a framebuffer update. Updates are triggered by the client which can autonomously decide when to request them. In addition to the screen, the RFB protocol supports any input device which can be mapped to a keyboard and a pointing device (e.g., a mouse) with multiple buttons. Besides these basic functions, VNC also provides different authentication and security methods, as well as extensions for better integration with the remote server (e.g., shared clipboard and file transfer).

RDP [28] is a proprietary protocol designed by Microsoft for remote input and display of hosts running the Windows operating system. It is based on the

Multipoint Application Sharing (T.128) recommendation by the International Telecommunication Union [29]. Despite being OS-specific, RDP clients and servers also exist for operating systems other than Microsoft Windows. RDP supports more primitives than VNC for screen updates as well as data transmission over multiple channels. It also interfaces with a wider range of input devices and provides several forms of redirection, including audio, printer, and filesystem access. Moreover, it supports authentication and built-in encryption.

GamingAnywhere [11] is an open source cloud gaming platform based on the client-server architecture and a modular design. Besides the input and audio/video capture modules, the system includes components for encoding multimedia data into streams that are delivered through the Real Time Streaming Protocol (RTSP). GamingAnywhere supports multiple platforms and employs H.264/AVC as the default format of video streams [30].

3.2. Characterizing Power Consumption of Mobile Devices

Since power consumption is a major issue in mobile devices, it is necessary to determine the actual expenditure accurately and efficiently. Besides energy modeling, several approaches rely on measurements to profile energy consumption with different levels of granularity.

The baseline approach consists in using a power meter attached to the battery of the mobile device. One of the most widely-used solutions in the literature is represented by the Monsoon Power Monitor [31], which is a complete hardware-software solution for measuring the power consumption of electronic devices. A special metering hardware allows a very accurate characterization of the power consumption, as the readings can be acquired with a high frequency (e.g., 5 kHz for the Monsoon power meter). However, as the meter is directly attached to the battery, this solution only provides a characterization of the system as a whole. As such, it cannot account for the power consumption due to individual applications and components such as the CPU, the radio and so on.

More recent research efforts have focused on how to build energy profiling

tools which can run on the mobile device and give a complete characterization of the different components involved. Among them, PowerTutor [32] is an Android application providing detailed measurements of the energy consumption in mobile devices. Specifically, it reports the power consumed by individual applications according to the different components involved – including CPU, display, network interface and GPS. Component and application-based profiling is realized through a model that estimates the power consumption based on different parameters. As the model was obtained and validated with a limited number of mobile devices, PowerTutor offers the best accuracy only with the supported platforms. Unfortunately, such platforms do not represent the current state-of-the-art in mobile device hardware. When used with platforms other than the supported ones, PowerTutor may not provide accurate power readings, or may not even work at all. There are also other software-based solutions such as AppScope [33] or the Trepro profiler [34]; however, they are either not available for download or not accurate enough for a reliable characterization of power consumption across multiple device types.

For all the reasons mentioned above, we use the Monsoon Power Monitor in our performance evaluation, as detailed in the following section.

4. Experimental Setup

We set up our experimental testbed according to the reference scenario illustrated in Figure 1. On the cloud side, we employed Amazon EC2 [35]. Specifically, we selected a G2 GPU instance running Microsoft Windows Server 2012 at the Amazon Europe West data center. On the mobile device side, instead, we employed a Samsung Galaxy S4+ smartphone (GT-I9506) equipped with a Long Term Evolution (LTE) transceiver and running Android 4.3. The mobile phone connected to the virtualized server through different wireless technologies. Specifically, we considered WiFi by employing a ZyXEL NBG5615 router directly connected to the university network. Moreover, we used LTE through the NetLeap 4G network provided by Nokia Solutions and Networks to Aalto

Element	Component	Description
<i>Smartphone</i> Samsung Galaxy S4+ (GT-I9506)	Platform	Quad-core 2.3 GHz CPU, 2 GB RAM
	WiFi	IEEE 802.11a/b/g/n/ac, dual-band
	LTE	Category 3 (100/50)
<i>Server</i> Amazon EC2 G2 GPU instance	CPU	8 Intel Xeon E5-2670 vCPUs, 15 GiB RAM
	GPU	NVIDIA, 1,536 CUDA cores, 4 GB vRAM
	Storage	60 GB SSD
<i>Wireless router</i> ZyXEL NBG5615	Standard	IEEE 802.11a/b/g/n, dual-band
	Data rate	300 Mbps (nominal)
<i>LTE network</i> NSN NetLeap	Frequency	2.6 GHz, 10 MHz channel
	Data rate	75 Mbps (nominal)

Table 1: Details of the different devices and technologies used in the testbed setup.

University for research and development purposes. The relevant characteristics of the devices and technologies used in our testbed setup are detailed in Table 1.

In all experiments, we set the brightness of the display to the maximum value allowed and disabled screen blanking to better characterize the related power consumption irrespective of the operating conditions (such as external light or user inactivity). In order to minimize the impact of other software running on the mobile device, we performed a factory reset before installing the applications used in the experiments and disabled all background traffic.

In order to characterize the performance of different remote access protocols, we considered the following software configurations.

- *Virtual Network Connection* (VNC). We used the RealVNC viewer for Android¹ at the mobile device and RealVNC Server² at the virtual instance.
- *Remote Desktop Protocol* (RDP). We used the 2X Client for Android³ at the mobile device and the RDP server shipped with Microsoft Windows Server 2012 at the virtual instance.
- *GamingAnywhere* (GA). We used the GamingAnywhere⁴ Android client

¹ <https://www.realvnc.com/products/android/>

² <http://www.realvnc.com/products/vnc/>

³ <http://www.2x.com/rdp-client/android/>

⁴ <http://gaminganywhere.org/download.html>

at the mobile device and the related server (version 0.7.5) at the virtual instance.

In order to evaluate the performance in realistic scenarios, we defined the following use cases, each corresponding to a practical usage of the mobile device for remote display access.

1. *Document editing and emailing.* In this use case, a user remotely connects to a desktop machine and generates an invoice for a customer based on a template that is customized for the specific occasion. Specifically, the user opens an existing Microsoft Word document, changes the quantity of the items, adds new items, calculates the new grand total, then generates a PDF file which is finally emailed to the customer. This scenario is motivated by the situation in which a salesman needs to take care of customer orders on the move.
2. *Remote software installation.* In this use case, a user connects to a remote desktop of another person to perform software installation or updates. Specifically, the user downloads the Adobe Reader XI software from the official website⁵, installs and configures it on the remote computer. This scenario is motivated by the situation in which an IT professional has to perform system maintenance or troubleshooting on a user workstation while out of office.
3. *Cloud gaming.* In this use case, a user connects to a virtual machine running the Trine 2 three-dimensional platform game⁶ and plays it remotely. Specifically, the user plays the first level of the game by running back and forth and jumping over obstacles when necessary. This scenario realizes an instance of mobile cloud gaming.

In order to have repeatable experiments that are not much affected by the timing of the user inputs, we have developed an Android application that records

---

⁵ <http://www.adobe.com/products/reader.html>

⁶ <http://trine2.com>

Figure 2: Measurement testbed used.

the touch events and replays them for a certain number of times. Even though we could not reuse the same recordings for different applications, due to differences in their user interface, we tried as much as possible to have a consistent sequence of touch events for all clients. We observed that the difference in the duration of the recordings was within ten seconds. We also verified that our software did not introduce unwanted bias in the power measurements – we found that it incurs in a CPU overhead that is less than one percent of the total power consumption on the average.

In the experiments, we set the resolution of the remote display to 1280 by 800 pixels and the color depth to the highest one supported by the considered software combinations (i.e., either 24 or 32 bits). In order to obtain information about the network utilization, we collected traffic traces at the server by using the Wireshark network protocol analyzer [36], similarly to [21].

We employed the Monsoon Power Monitor [31] to measure the power consumption of the mobile device. To this end, we removed the battery of the smartphone and attached the meter directly to the corresponding contacts on the motherboard. We then connected the meter to a workstation that logged and processed the power consumption readings. The measurement setup is illustrated in Figure 2.

In our evaluation, we replicated each experiment five times. We then derived

Wireless Connection	Round-trip-time (ms)				Data rate (Mbps)	
	Min	Max	Avg	Stdev	Downlink	Uplink
WiFi	46.32	153.61	77.55	10.55	37.91	47.87
LTE	59.21	105.06	60.76	2.26	62.30	20.51

Table 2: Round-trip-time and data rate of the wireless connections in the experimental setup.

the average values along with the corresponding 90% confidence intervals, shown as error bars in the figures.

5. Experimental Results

In this section, we provide the results obtained from our experiments with focus on power consumption and network utilization. We conclude the section with a discussion of the user experience and summary considerations.

5.1. Performance Evaluation

In the experiments, we evaluated the following metrics.

- *Average power consumption*, obtained by averaging the instantaneous power consumption reported by the meter over the whole duration of an experiment.
- *Bandwidth utilization*, expressed in terms of the total number of bytes exchanged during an experiment divided by the corresponding duration.

We performed experiments for the different protocols and the use cases introduced in the previous section. Furthermore, we considered both WiFi and LTE as wireless communication technologies.

In order to support the analysis of the experimental results, we run some benchmarks to characterize the performance of the considered wireless networks. The obtained results are summarized by Table 2. In detail, the table shows the round-trip-time (RTT) between the smartphone and the cloud server for the two wireless connections. We obtained the RTT by running the ping tool right before starting the experiments; ping messages were generated every second

Figure 3: (a) Average power consumption and (b) bandwidth utilization as a function of the considered protocols for different wireless communications technologies in use case 1.

and lasted for 10 minutes. The table also shows the data rate measured at the smartphone with the Speedtest.net network testing tool.

Figure 3 shows the performance of the considered protocols as a function of the wireless technology for use case 1. Specifically, Figure 3a shows the average power consumption of the smartphone. The results show that the two remote desktop protocols (i.e., VNC and RDP) obtain a lower power consumption than GA, even though the difference is not very significant (within 15% for both wireless connections). RDP is the protocol that achieves the lower power consumption in this scenario. All protocols show a consistent trend, i.e., the power consumption for LTE is higher than that for Wi-Fi (about 30% on the average). This is not surprising as cellular network access is largely affected by the so-called tail energy, i.e., the energy spent in the high-power states that immediately follow the completion of a data transfer [37].

Figure 3b shows the uplink and downlink bandwidth utilization of the considered solutions for the different wireless connections and allow to relate the power consumption to the network traffic. It is apparent how GA generates a much higher network traffic than VNC and RDP; the difference is about one order of magnitude (e.g., 1 Mbps versus 100 Kbps). This is due to the fact

that GA is continuously streaming the content of the captured video and the generated traffic is not much affected by the actual changes in the content of the remote screen. Conversely, both VNC and RDP use a different approach, i.e., they find the rectangular region delimiting the content of the screen that has changed. As use case 1 involves a typical desktop scenario, the interactions with the remote server result in limited changes in the content of the screen that can be efficiently encoded into rectangular screen updates. The higher power consumption of VNC over RDP is also explained through the higher amount of generated traffic.

Quite surprisingly, the power consumption of GA does not seem to be significantly affected by the bandwidth utilization. Even though it is difficult to evaluate precisely the impact of the different factors affecting the power consumption of the radio (e.g., the related power management policies), it is clear how the continuous data stream is affecting the performance. Another aspect to consider is the decoding of the remote screen at the client. Both VNC and RDP send screen updates from the server to the client as compressed bitmaps, whereas GA sends a video stream encoded in H.264/AVC format. The decoding process is more complex and power-hungry for such a video stream rather than for regular images; however, the GA client exploits hardware-based acceleration which is more efficient than the software version. This explains why the power consumption for GA is not extremely higher than that of VNC and RDP.

Figure 4 shows the performance of the considered protocols as a function of the wireless technology for use case 2. The related power consumption and bandwidth utilization are detailed in Figures 4a and 4b, respectively. The two figures overall show a similar trend than that obtained for use case 1. RDP is the protocol that achieves both the lowest power consumption and the lowest bandwidth utilization. From the network perspective, we observe that there is a lower uplink traffic for all solutions if compared to use case 1. This is highly related to the nature of the considered use case in which less input is generated by the user; most of the session is spent by waiting for the installation process to complete and by verifying that the software is working as intended. The limited

Figure 4: (a) Average power consumption and (b) bandwidth utilization as a function of the considered protocols for different wireless communications technologies in use case 2.

amount of input is reflected by the reduced amount of uplink traffic conveying control messages to the server in both VNC and RDP. This reduction is more significant for RDP, which also explains the corresponding lower power consumption. In this use case, VNC generates slightly more traffic when used over WiFi rather than over LTE. This happens as the WiFi connection we employed was subject to interference and the protocol appears to be rather sensitive to message losses, especially when the related retransmissions affect a large framebuffer update. Indeed, the client we used periodically requested framebuffer updates even in absence of any user input in this specific use case. On the other hand, GA keeps generating the same uplink traffic as in use case 1 as a result of the design criteria specifically targeted to cloud gaming scenarios [11].

Figure 5 shows the performance of the considered protocols as a function of the wireless technology for use case 3. In detail, Figure 5a shows the average power consumption of the smartphone. We can observe how the results in the figure are radically different from those obtained for the other use cases. In fact, GA has the lowest power consumption (except for VNC with WiFi) in this scenario. A similar trend is shown by Figure 5b, in which the bandwidth utilization of both VNC and RDP are one order of magnitude higher than that

Figure 5: (a) Average power consumption and (b) bandwidth utilization as a function of the considered protocols for different wireless communications technologies in use case 3.

of GA. It is apparent how protocols specifically designed for remote desktop access cannot cope with the highly dynamic content of the remote display while the user is playing the game. Even though this result is not unexpected, the magnitude of traffic generated by VNC and RDP is remarkably high. There is also a significant difference between the values obtained with LTE and Wi-Fi. Specifically, the bandwidth utilization is much lower for Wi-Fi than for LTE. This happens because the remote desktop protocols cannot cope with the amount of screen updates and lose many frames especially for the (slower) Wi-Fi connection. Our observation is consistent with the results reported in [22], even though there are differences in the considered setups. This also explains why VNC obtains the lowest power consumption under Wi-Fi.

We finally discuss the average power consumption as a function of time (expressed as a percentage of the session duration) for the considered remote desktop protocols under use case 2 and different radio communication technologies (Figures 6 and 7). All samples logged by the meter have been grouped with a granularity of 0.5 seconds. The figures show the average value of grouped samples (i.e., the dark line) as well as the corresponding minimum and maximum values (i.e., the light gray bands).

Figure 6: Average power consumption as a function of time in a representative session of use case 2 under WiFi for (a) VNC and (b) RDP.

We now focus on the results obtained for RDP and VNC with a WiFi connection. Figure 6 clearly highlights how RDP is less bursty than VNC, even though the difference between the highest and lowest power consumption is consistently high throughout the session. In contrast, VNC has several peaks in the power consumption, but the variance of the minimum and maximum values is higher. In other words, the power consumption of VNC better follows the number of events in the session. For instance, we can see from Figure 6a that most of the peaks occur at the beginning and at the end of the session, while the power consumption is consistently lower and more regular in the range between 20% to 80% of the total time. It is worth noting that, in use case 2, most of the user-triggered events occur at the beginning (i.e., for accessing the software website to initiate its download) and at the end of the session (i.e., for testing that the application has been correctly installed). For the rest of the time, the user is mostly idle while waiting for the download and installation process to proceed. Thus, there are only a few events produced by the user at the mobile device, and most of the screen updates that are not directly related to them (e.g., the display of the installation progress and the related information).

Figure 7: Average power consumption as a function of time in a representative session of use case 2 under LTE for (a) RDP and (b) GA.

Figure 6b shows that, instead, RDP results in a higher power consumption irrespective of the actual inputs and screen updates. Apparently, the used software keeps exchanging messages periodically, even in absence of user inputs and screen updates. This justifies the much higher consumption of RDP in use case 2 when compared to VNC, as shown by Figure 4a, and is consistent with the traffic data in Figure 4b.

Lastly, we consider the results obtained for RDP and GA with an LTE connection (Figure 7). Figure 7a clearly shows that the power consumption under LTE is much higher than the corresponding one obtained with WiFi (Figure 6b). Besides, the figure shows a radically different trend. In fact, the spread of the power consumption is remarkably higher in this case, with a higher difference between the minimum and maximum values reported. Moreover, the average itself has a much higher variability, especially in the range between 15% and 35% of the session duration, while it is consistently low between 40% and 50% of the session duration. This is due to the power management scheme adopted in LTE and, specifically, the state transitions which are highly affected by the timing between subsequent network events, clearly showing the factor

caused by the tail energy [37]. Similar considerations hold for VNC over LTE, whose results are not reported here for the sake of conciseness.

The power consumption of GA is consistently high because the desktop content is continuously streamed to the client, as illustrated in Figure 7b. As a consequence, different subsystems in the smartphone (e.g., microcontroller, graphics card and radio transceiver) are simultaneously active for a longer time. This reduces the spread between the maximum and minimum power consumption. Rather interestingly, the actual maximum values of the power consumption are lower than those in Figure 7a. This relates to the lower number of switches between different power management states, ultimately resulting in an average power consumption closer to the minimum values reported. Indeed, GA exhibits a similar trend under WiFi, with an average power consumption around 2.2 W, which is consistent with the results reported in Figure 4a.

5.2. Additional Considerations and Summary

Besides characterizing the energy efficiency of the different remote display protocols, the experiments allowed us to evaluate the different solutions in terms of usability and user experience. Even though the findings are related to the actual software configuration employed, we believe that they are representative of the technical features behind the used protocols.

All of the tested remote access solutions proved to be responsive enough in basic desktop usage. Use cases 1 and 2 were easy to accomplish with all client/server combinations. There were no noticeable response delays with neither WiFi nor LTE. All three solutions emulated the mouse movements and VNC and RDP also had a full keyboard emulation mode. Instead, we had to add some text entering capabilities ourselves to GA in order to accomplish the use cases. The keyboard emulation mode of the 2X Client (RDP) was the most usable, since it did not fill the whole screen of the mobile phone when entering text. This is important, as the user can instantly check for typing errors. VNC Viewer uses the standard Android keyboard input with special keys added; as a consequence, it fills the screen almost entirely, thus making it hard to write

1
2
3
4
5
6
7
8
9 long texts. The 2X Client also had the best mouse emulation mode thanks to
10 an option of a virtual movable mouse on the screen. On the other hand, both
11 the 2X Client and the VNC Viewer lacked the option to lock the viewable area
12 of the screen: they had to be constantly centered during the tests. GA better
13 handled this aspect, as it showed the whole display full-screen without the pos-
14 sibility of accidentally resizing the viewport. Nevertheless, VNC and RDP are
15 still the best choices for basic desktop usage because they consumed less energy
16 and bandwidth in use cases 1 and 2.

17
18 As for use case 3, GA showed its real potential, while VNC together with
19 RDP revealed their weaknesses. VNC and RDP are not very suitable for situa-
20 tions where a large portion of the remote display changes rapidly, such as during
21 video playback or, indeed, gaming. Clearly, VNC was not designed for such an
22 usage, since it does not even have remote audio functionalities. Note that the
23 gaming use case still applies to 3D modeling and other graphic-intensive tasks.
24 RDP was slightly better than VNC in use case 3, but the game was still not
25 playable. We also observed that the drivers of the graphics adapter cannot be
26 directly accessed when RDP is used. This limits the computing power of the
27 remote server. GA and VNC can use the full potential of the GPU; however,
28 the encoding schemes used in VNC are not powerful enough for rich graphics.
29 The simultaneous strength and weakness of GA is that its bandwidth usage and
30 responsiveness do not vary depending on the display change pattern. This is
31 good for the gaming use case, but there is excessive overhead compared to VNC
32 and RDP for simple desktop usage scenarios.

33
34 We would like to stress that a quantitative evaluation of the usability for the
35 different solutions considered here would be highly dependent on the scenario
36 and the software used, thus being out of the scope of this article; the interested
37 reader may refer to [14] and [23] for a related discussion. However, based on
38 the measurements and the observations presented in this article, it is clear that
39 there is room for improving remote access solutions for mobile devices, with
40 reference to both protocol and usability aspects. For instance, none of the
41 tested implementations supported sensors or gestures as controls. It would be

also beneficial to design a hybrid solution capable to adapt the remote access protocol on-the-fly, depending on the characteristics of the displayed content. Such a hybrid approach could, for instance, switch to using fully compressed video encoding when there are changes in the display over a certain threshold and switch back to VNC or RDP otherwise. It may also be possible to apply the video encoding schemes used in GA and similar solutions only for the portions of the screen that are changing rapidly.

6. Conclusion

In this article, we evaluated the performance of remote display access as a means for mobile cloud computing. To this end, we considered realistic use cases, including office productivity, remote assistance, and gaming. Moreover, we analyzed different remote access solutions, i.e., Virtual Network Connection (VNC), Remote Desktop Protocol (RDP), and GamingAnywhere (GA). We performed extensive experiments involving virtual machines deployed in the cloud and accessed through both WiFi and Long Term Evolution. We focused on power consumption and bandwidth utilization, but also provided some considerations on usability and user experience. Our results have shown that GA has a much higher bandwidth utilization than VNC and RDP as well as a higher power consumption. Such a power consumption, however, is not significantly higher as one would expect, as the protocol and its implementation are optimized for mobile devices. VNC and RDP overall utilize a very low bandwidth for sessions with a limited number of events and screen updates. However, they cannot cope with highly dynamic screen content, especially in terms of satisfactory user experience.

As a future work, we are considering to extend GamingAnywhere in order to make it more suitable for desktop scenarios other than gaming. We are also planning to model the energy consumption of remote access on mobile devices in the specific context of cloud computing.

Acknowledgments

We are grateful to the anonymous reviewers for their insightful comments and constructive suggestions that allowed us to significantly improve the quality of the manuscript.

This work has been partially supported by the US National Science Foundation (NSF) under grant CNS-1150192 and by the Academy of Finland under grant number 278207. Any opinions, findings and conclusions or recommendations expressed in this article are those of the authors and do not necessarily reflect those of the NSF.

References

- [1] K. Pentikousis, In search of energy-efficient mobile networking, *Communications Magazine, IEEE* 48 (1) (2010) 95–103. doi:10.1109/MCOM.2010.5394036.
- [2] E. Cuervo, A. Balasubramanian, D.-K. Cho, A. Wolman, S. Saroiu, R. Chandra, P. Bahl, MAUI: making smartphones last longer with code offload, in: *Proc. of MobiSys '10*, 2010, pp. 49–62. doi:10.1145/1814433.1814441.
- [3] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, A. Patti, CloneCloud: elastic execution between mobile device and cloud, in: *Proc. of EuroSys '11*, 2011, pp. 301–314. doi:10.1145/1966445.1966473.
- [4] K. Kumar, Y.-H. Lu, Cloud computing for mobile users: Can offloading computation save energy?, *Computer* 43 (4) (2010) 51–56. doi:10.1109/MC.2010.98.
- [5] V. Nambodiri, T. Ghose, To cloud or not to cloud: A mobile device perspective on energy consumption of applications, in: *World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2012 IEEE International Symposium on a, 2012, pp. 1–9. doi:10.1109/WoWMoM.2012.6263712.

- [6] S. J. Yang, J. Nieh, M. Selsky, N. Tiwari, The performance of remote display mechanisms for thin-client computing, in: USENIX Annual Technical Conference, General Track, 2002, pp. 131–146.
- [7] P. Simoens, F. De Turck, B. Dhoedt, P. Demeester, Remote display solutions for mobile cloud computing, *Computer* 44 (8) (2011) 46–53. doi:10.1109/MC.2011.70.
- [8] Y. Lu, S. Li, H. Shen, Virtualized screen: A third element for cloud-mobile convergence, *Multimedia, IEEE* 18 (2) (2011) 4–11. doi:10.1109/MMUL.2011.33.
- [9] R. A. Baratto, S. Potter, G. Su, J. Nieh, MobiDesk: mobile virtual desktop computing, in: *Proc. of MobiCom '04*, 2004, pp. 1–15. doi:10.1145/1023720.1023722.
- [10] C.-L. Tsao, S. Kakumanu, R. Sivakumar, SmartVNC: an effective remote computing solution for smartphones, in: *Proc. of MobiCom '11*, ACM, New York, NY, USA, 2011, pp. 13–24. doi:10.1145/2030613.2030616.
- [11] C.-Y. Huang, K.-T. Chen, D.-Y. Chen, H.-J. Hsu, C.-H. Hsu, GamingAnywhere: The first open source cloud gaming system, *ACM Trans. Multimedia Comput. Commun. Appl.* 10 (1s) (2014) 10:1–10:25. doi:10.1145/2537855.
- [12] A. Lai, J. Nieh, Limits of wide-area thin-client computing, *SIGMETRICS Perform. Eval. Rev.* 30 (1) (2002) 228–239. doi:10.1145/511399.511363.
- [13] A. M. Lai, J. Nieh, On the performance of wide-area thin-client computing, *ACM Trans. Comput. Syst.* 24 (2) (2006) 175–209. doi:10.1145/1132026.1132029.
- [14] N. Tolia, D. Andersen, M. Satyanarayanan, Quantifying interactive user experience on thin clients, *IEEE Computer* 39 (3) (2006) 46–52. doi:10.1109/MC.2006.101.

- [15] S. Kosta, A. Aucinas, P. Hui, R. Mortier, X. Zhang, ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading, in: Proc. of INFOCOM 2012, 2012, pp. 945–953. doi:10.1109/INFCOM.2012.6195845.
- [16] A. P. Miettinen, J. K. Nurminen, Energy efficiency of mobile clients in cloud computing, in: Proc. of HotCloud '10, 2010, pp. 4–4.
- [17] E. Chen, M. Itoh, Virtual smartphone over IP, in: Proc. of WoWMoM 2010, 2010, pp. 1–6. doi:10.1109/WOWMOM.2010.5534992.
- [18] L. Deboosere, B. vankeirsbilck, P. Simoens, F. De Turck, B. Dhoedt, P. Demeester, Cloud-based desktop services for thin clients, Internet Computing, IEEE 16 (6) (2012) 60–67. doi:10.1109/MIC.2011.139.
- [19] T. Richardson, Q. Stafford-Fraser, K. Wood, A. Hopper, Virtual network computing, Internet Computing, IEEE 2 (1) (1998) 33–38. doi:10.1109/4236.656066.
- [20] B. Joveski, M. Mitrea, P. Simoens, I. J. Marshall, F. Prêteux, B. Dhoedt, Semantic multimedia remote display for mobile thin clients, Multimedia Systems 19 (5) (2013) 455–474. doi:10.1007/s00530-013-0304-6.
- [21] C.-Y. Huang, C.-H. Hsu, Y.-C. Chang, K.-T. Chen, GamingAnywhere: An open cloud gaming system, in: Proceedings of the 4th ACM Multimedia Systems Conference, MMSys '13, 2013, pp. 36–47. doi:10.1145/2483977.2483981.
- [22] Y.-C. Chang, P.-H. Tseng, K.-T. Chen, C.-L. Lei, Understanding the performance of thin-client gaming, in: Communications Quality and Reliability (CQR), 2011 IEEE International Workshop Technical Committee on, 2011, pp. 1–6. doi:10.1109/CQR.2011.5996092.
- [23] M. Jarschel, D. Schlosser, S. Scheuring, T. Hossfeld, An evaluation of QoE in cloud gaming based on subjective tests, in: Innovative Mobile and In-

- ternet Services in Ubiquitous Computing (IMIS), 2011 Fifth International Conference on, 2011, pp. 330–335. doi:10.1109/IMIS.2011.92.
- [24] T. Kämäräinen, M. Siekkinen, Y. Xiao, A. Ylä-Jääski, Towards pervasive and mobile gaming with distributed cloud infrastructure, in: The 13th Annual Workshop on Network and Systems Support for Games (NetGames 2014), 2014, pp. 1–6. doi:10.1109/NetGames.2014.7008957.
- [25] H. Li, P. Li, S. Guo, X. Liao, H. Jin, Modeap: Moving desktop application to mobile cloud service, *Mobile Networks and Applications* 19 (4) (2014) 563–571. doi:10.1007/s11036-014-0518-7.
- [26] Y. Lin, M. Di Francesco, Energy consumption of remote desktop access on mobile devices: An experimental study, in: The 1st IEEE International Conference on Cloud Networking (CloudNet 2012), 2012. doi:10.1109/CloudNet.2012.6483663.
- [27] T. Richardson, J. Levine, The Remote Framebuffer Protocol, RFC 6143, Internet Engineering Task Force (IETF) (March 2011).
URL <http://www.ietf.org/rfc/rfc4627.txt?number=6143>
- [28] Microsoft Corp., Remote Desktop for Mobiles, <http://msdn.microsoft.com/en-us/library/aa383015.aspx> (retrieved July 6, 2012).
- [29] Telecommunication Standardization Sector of ITU, Multipoint application sharing, Recommendation T.128, International Telecommunication Union (ITU) (June 2008).
URL <http://www.itu.int/rec/T-REC-T.128-200806-I/en>
- [30] T. Wiegand, G. J. Sullivan, G. Bjontegaard, A. Luthra, Overview of the H.264/AVC video coding standard, *Circuits and Systems for Video Technology, IEEE Transactions on* 13 (7) (2003) 560–576. doi:10.1109/TCSVT.2003.815165.

- 1
2
3
4
5
6
7
8
9 [31] Monsoon Power Solutions Inc., Mobile Device Power Monitor Manual
10 – Version 1.8, [http://msoon.github.com/powermonitor/PowerTool/doc/](http://msoon.github.com/powermonitor/PowerTool/doc/Power%20Monitor%20Manual.pdf)
11 [Power%20Monitor%20Manual.pdf](http://msoon.github.com/powermonitor/PowerTool/doc/Power%20Monitor%20Manual.pdf) (June 2012).
12
13
14 [32] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, L. Yang,
15 Accurate online power estimation and automatic battery behavior based
16 power model generation for smartphones, in: Proc. of CODES/ISSS '10,
17 2010, pp. 105–114. doi:10.1145/1878961.1878982.
18
19 [33] C. Yoon, D. Kim, W. Jung, C. Kang, H. Cha, Appscope: Application
20 energy metering framework for android smartphone using kernel activity
21 monitoring, in: USENIX Annual Technical Conference, 2012, pp. 387–400.
22
23 [34] Qualcomm Technologies Inc., Trepn Profiler, [https://developer.](https://developer.qualcomm.com/develop/development-devices/trepn-profiler)
24 [qualcomm.com/develop/development-devices/trepn-profiler](https://developer.qualcomm.com/develop/development-devices/trepn-profiler), retrieved
25 March 13, 2015.
26
27 [35] Amazon Inc., Amazon Elastic Compute Cloud, [http://aws.amazon.com/](http://aws.amazon.com/ec2/)
28 [ec2/](http://aws.amazon.com/ec2/).
29
30
31
32 [36] A. Orebaugh, G. Ramirez, J. Burke, L. Pesce, Wireshark & Ethereal Net-
33 work Protocol Analyzer Toolkit (Jay Beale's Open Source Security), Syn-
34 gress Publishing, 2007.
35
36 [37] N. Balasubramanian, A. Balasubramanian, A. Venkataramani, Energy con-
37 sumption in mobile phones: A measurement study and implications for
38 network applications, in: Proceedings of the 9th ACM SIGCOMM Con-
39 ference on Internet Measurement Conference, IMC '09, 2009, pp. 280–293.
40 doi:10.1145/1644893.1644927.
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65