Thin is Green: Leveraging the Thin-Client Paradigm for Sustainable Mobile Computing

Toolika Ghose, Vinod Namboodiri, Ravi Pendse
Department of Electrical Engineering and Computer Science
Wichita State University, Wichita, KS, USA
Email: {txghose, vinod.namboodiri, ravi.pendse}@wichita.edu

Abstract—Progress in the area of environmental sustainability for the mobile computing industry could be achieved by making advancement on two fronts: reducing the energy consumed by individual devices throughout their life cycle and reducing the rate at which these devices are discarded. In this work, to address both fronts, we propose the use of a thin-client approach, whereby a mobile device relies mainly on the resources at a remote server to carry out application tasks. To assess the benefits of the proposed approach, this work develops an analytical model as well as performs an empirical evaluation of performance and energy consumption on Android-based smartphones. In terms of energy, a reduction of approximately 11% in the average life cycle energy (LCE) is seen by increasing the device’s usage life by even three months through a thin-client approach. In terms of performance, a thin-client device is shown to improve execution by 57% compared to a self-reliant device.

Keywords—Green computing; reusability; sustainability; thin client; life cycle energy; power saving

I. INTRODUCTION

The exponentially increasing demand for smartphones, tablets, and laptops has transformed the mobile communications sector into one of the fastest growing industries. The first phone call over the Global System for Mobile Communications (GSM) occurred in 1991, and presently 77% of the global population uses cell phones [1]. About 174 million smartphones were sold globally in 2009. It has been forecasted that smartphone sales will increase at a compounded rate of 30% by the year 2016. The ANALYST foresees that smartphones would contribute to a $320 billion market by 2016 [2]. In 2011, there were nearly five billion global mobile users, and 85% of these devices were capable of accessing the World Wide Web. The progression in mobile device technology also has brought a great deal of mobility and global connectivity. Accessing the Web and social networks through smartphones is becoming popular day by day. It is known that 61% of mobile subscribers in the United States have smart mobile devices [3].

The features and functionality of these mobile devices improve with time, but unfortunately their lifespan is very short, with most users upgrading their devices every 18 to 24 months [4]. This short lifespan leads to millions of cell phones and other mobile devices being discarded every year, thus contributing to a significant part of electronic waste, or e-waste. It was reported in [5] that nearly 19,500 tons of mobile devices were discarded in 2010, of which merely 2,240 tons were recycled and the remaining 17,200
tons contributed to e-waste. Additionally, the increase in features and functionality of these smart devices is also responsible for a significant increase in their global energy consumption. Of the total energy consumed by information and communication technology (ICT), mobile devices are responsible for 10–20% [1, 6, 7].

In view of developing a reliably green and sustainable solution, we propose a model for reutilization of old mobile devices with the help of a cloud-based thin-client approach. This thin-client paradigm primarily operates on the concept of offloading a majority of computations to a remote server for executing the application logic. In most cases, the client is used for displaying the graphical output only. In this work, we propose an analytical model based on various critical parameters, such as life cycle energy (LCE), or energy consumption at various levels of the life cycle, performance improvement of old mobile devices, etc. We further evaluate the performance improvement, end-user quality of the experience, and energy savings that could be achieved by reusing old or discarded mobile phones under a thin-client paradigm. The novelty of our work is accentuated via the LCE model for mobile devices, which provides a comprehensive insight into where and how energy savings could be achieved in the device’s overall life cycle. Additionally, the life cycle model provides a clear path for steering the design towards a fully sustainable structure.

The specific contributions of this work include the following:

- A proposal for and evaluation of an analytical model for LCE consumption of mobile devices in order to understand energy consumption during various phases of a mobile device’s life cycle.
- A derivation of the theoretical conditions for energy savings under a thin-client paradigm.
- A comparative empirical evaluation of performance and energy consumption of old and new Android-based smartphones operating under the thin-client paradigm.

Results of this work provide insight into the impact on LCE consumption of mobile devices due to an increase in number of usage phases. It is shown that a reduction in the average LCE of approximately 11% can be achieved with just a three-month increase in the duration of a device’s usage phase. The proposed thin-client approach for smartphone reutilization was experimentally implemented by means of a laboratory-based setup of the thin-client scenario. Through our experiments, we show that an older device using the thin-client approach could execute tasks 90% faster than a newer device with greater capabilities.

The remainder of this paper is organized as follows: Section II presents related work. Section III presents an analytical model for a mobile device over its life cycle and associated numerical evaluations examining the relation between energy consumption and reutilization. Section IV derives conditions for a device using the thin-client approach to save energy compared to a conventional device. The methodology employed for empirical evaluations of energy savings and performance of devices
employing the thin-client approach is then described in Section V followed by the results discussed in Section VI. Finally, Section VII presents the conclusions of this work followed by a discussion of future work.

II. BACKGROUND

The ongoing expansion in the number of smart mobile devices has created multiple environmental challenges such as e-waste as a consequence of a device’s shorter lifespan and a significant increase in global energy consumption. The driving factor for a shorter lifespan of 18 to 24 months is due to advancement in the technology and telecom provider contract plans. For instance, some of the largest telecom providers in the United States are now introducing newer mobile phone contracts where users can upgrade their smartphone devices after just one year of usage [8]. On the other hand, the central processing unit (CPU) processing power and mass storage capacity doubles every one and a half years, which also drives consumers to opt for newer devices [1].

Important factors such as features, capabilities, and upgrade opportunities, as discussed above, are some of the major driving elements for the short lifespan of smart mobile devices. However, a small percentage of these discarded devices are recycled and the rest (nearly 75%) contribute to electronic waste [5, 9]. An additional impact on the global environment and human health comes from several toxic materials that are present in most of these mobile devices, including arsenic, lead, mercury, cadmium, nickel, copper, and zinc [9, 10].

Although features and functionality of smartphones have significantly improved with time, these devices still lack processing capabilities, compared to desktops, and they have a limited battery life. A consequent upsurge in features and functionality of these devices has also led to greater levels of energy consumption. Such an increase in energy consumption impacts user productivity and convenience through faster depletion of the limited battery life. The collective global energy consumption by mobile devices is another major characteristic of the mobile communications domain. This type of consumption has increased proportionally with proliferation in features and functionality. Hence, the creation of a more energy-efficient model of mobile devices has been a primary focus for many researchers in the mobile communications sector.

In this paper, we propose a thin-client-server approach for mobile devices to make these devices more energy efficient in order to maximize their battery lifetime. An extension of the power-consumption model we proposed in [16] via energy savings by mobile devices during the usage phase under a thin-client paradigm is presented in the next sub-section. A thin-client-server model for mobile devices is primarily based on its operating in conjunction with a remote server for resource sharing, accessibility of features, and functionality [11].

Mobile thin clients are simplified devices with considerably fewer hardware components than traditional mobile devices. The thin-client-server model focuses primarily on enhancing the capability of mobile devices by providing full functionality to the
device through a connection to remote servers. These devices could be compared to the “dumb” terminals during the mainframe era in terms of the limited features and hardware capabilities of both machines. Thin clients provide several benefits over traditional mobile devices through access transparency based on location due to mobility, an increase in personal data storage, resource-intensive applications executed on the cloud, and ease of maintenance and upgrading [12, 13]. The thin-client-server approach is a technology that is being adapted by industry to reduce manufacturing and maintenance costs of the devices while effectively trying to meet user requirements in terms of performance and reliability. Thin clients are becoming prevalent due to the low cost of the required hardware and increased lifespan of these devices. A noteworthy feature of the thin-client-server model is the capability to decisively perform an application execution either locally on the mobile device or remotely on the server, based on the availability of resources and complexity of the required computation. Thus, the thin-client-server model provides a major advantage by promoting energy-efficient usage on the local mobile device by diverting computation-intensive tasks to a remote location. Thin-client desktops can lower the maintenance and upgrade costs by 50–70%. The downtime cost is more for thin-client architecture, due to the advanced client-server infrastructure. Research performed by Forrester Research Inc. in 2008 shows that thin clients reduce power usage by 24% and carbon dioxide emissions by 23%, and increase the lifespan of desktops by five to ten years [13].

Miettinen and Nurminen [14] report that computation offloading can be done to improve performance and save energy under the condition that the workload consists of more than 1,000 cycles of computation for each byte of data. Portable devices using a wireless local area network (WLAN) for remote connectivity were reported to consume less energy than third generation (3G) devices, because 3G has more communication latencies. The work in [14] aims at determining the critical factors affecting energy consumption of mobile clients in cloud computing and also aims at comparing the relative impact of communication and local processing. However, that study is limited to an empirical approach without an analytical characterization. The work in [15] used an analytical model to analyze the energy savings through offloading computations. The work in [16, 17, 18] consider a similar problem from a system-level perspective, identifying the conditions when individual methods should be offloaded to a remote server to save energy. The authors in [16] proposed a similar framework called MAUI for offloading that considered the energy consumption in transferring or downloading the computational work or application program to a remote server. MAUI does the profiling of applications, whereby the profiler runs continuously to determine the cost of each method in order to make the offloading decision. A local as well as remote copy of the same application program is always maintained on the machine. It uses a linearization technique to calculate the networking and CPU costs of offloading, considering the network connectivity [16]. The work in [17, 18] also approached the problem we consider from a system-level perspective, identifying opportunities when code and its individual methods should be offloaded to a server in order to save energy. In contrast, this work takes an application-level
approach that considers an application as a single unit that cannot be decomposed into multiple methods and must be run either on the cloud or locally. We feel this approach is justified because most existing applications are designed in this fashion to run either on the cloud or locally, but not both. Typically partitioning and profiling methods of offloading techniques can be too complex and increase the run time for simple applications. This work proposes a simpler and easy-to-implement friendly model rather than employing complex partitioning and profiling methods. In contrast, our current work considers an application-level approach because the existing applications are designed to be executed either locally or on the cloud, but not both. The work in [14, 15, 16, 17, 18] proposes different offloading techniques; however, these energy models are limited in order to optimize the use-phase energy consumption for mobile devices, whereas the work in this paper focuses on energy savings of the overall life cycle of mobile devices. The current work also considers offloading as one technique to increase the lifespan of mobile devices and uses it to compare the performance of phones with different hardware capabilities.

Although thin clients have been broadly evaluated in the literature for their performance and user experience for desktop scenarios, they rarely have been analytically studied via a mobile computing paradigm approach that is not only green in terms of reducing e-waste but also prudent in energy consumption. Using a cloud-based thin-client paradigm, hardware upgrades can be performed at the server side, thus reducing the number of devices that are disposed of each year. Portable devices lack processing power and have limited battery life. By using thin-client-server architecture, resource-poor portable devices can be used as powerful computers. The cloud-based approach can potentially resolve some constraints of portable devices, such as storage, memory, processing power, and battery life.

The work in [11] solely focused on the analytical study of the power consumption of mobile devices used under a thin-client paradigm and included a theoretical energy model for any application execution under a thin-client paradigm, whereas our current work described in this manuscript analyzes energy consumption during various phases of a mobile device’s life cycle. We also perform a comparative study of energy consumption of Android-based old and new smartphones operating under the thin-client paradigm. The current work considers offloading as one of the techniques to increase the lifespan of mobile devices. The primary focus of [32] was to compare the energy consumption behavior for large and small form factor devices. The authors also propose a GreenSpot algorithm to save energy on small form factor devices in [19]. In contrast, this paper proposes the use of a thin-client approach as one possible solution to reducing the energy consumption of mobile devices while reducing their rate of replacement. In order to assess energy consumption through the lifespan of a mobile device, this work develops an analytical model. Furthermore, to assess the practical feasibility of using a thin-client approach, an empirical evaluation of performance and energy consumption is done on Android-based smartphones.
In this paper, we address the issue of developing plausible techniques to reduce the life cycle energy consumption of mobile devices. We introduce one such green and environmentally sustainable approach for mobile devices by leveraging the thin-client paradigm. A thin-client paradigm for smart mobile devices could provide progress on environmental sustainability for the mobile computing paradigm achieved by making progress on two fronts: reducing the energy consumed by each individual device (including energy costs to manufacture them), and reducing the rate at which these devices are discarded. We analyze the LCE consumption of mobile devices and provide insight on where energy savings could be achieved throughout the life cycle. We argue that a reduction in the average LCE of mobile devices could be achieved under a thin-client paradigm. We also analyze the performance improvement of mobile devices when reused under a cloud-based thin-client paradigm.

III. MOBILE DEVICE LIFE CYCLE ENERGY CONSUMPTION

The following section discusses the analytical characterization via the proposed LCE consumption model. The model was designed to be generic enough so that it could be implemented on any mobile device. The goal of the proposed analytical model is to identify the impact of increase in number of usage stages on the overall LCE cost of a mobile device. In the subsequent section, this model will be used to evaluate the performance and energy consumption of devices such as smartphones when reused under a thin-client paradigm.

A. Proposed Model for Energy Consumption Analysis

In this model, it was assumed that the total energy consumed throughout the life cycle of a mobile device consists of three major components: manufacturing, usage, and recycling. Consistent with our assumptions, three major life cycle phases were considered in our theoretical model along with corresponding phase energies defined as manufacturing energy ($e_m$), energy consumed during the usage phase ($e_u$), and recycling energy ($e_r$). Additionally, it was assumed that mobile devices could be reused. Moreover, the reutilization of these devices could also consist of several usage stages ranging from 1 to $n$ as shown in Figure 1. The parameters that define the model are shown in Table 1.

![Figure 1. Model considered for sustainable mobile devices](image)

Table I. Definitions of parameters used in model
Each device begins with a manufacturing phase that is associated with manufacturing energy \((e_m)\). The device then traverses through various usage stages in the life cycle based on the probability \((p)\) to exit to the recycling phase after completion of each usage phase. It was assumed that each device mandatorily enters the first usage phase, as shown in Figure 1. The recycling phase is associated with a recycling energy \((e_r)\), which also contributes to the total LCE consumption.

Consider that the probability to exit to the recycling phase is \(p\), and \(n\) is the number of independent usage phases that the device traversed before exiting into the recycling phase at the \(nth\) usage phase. Thus, the probability mass function of any device exiting to the recycling phase at the \(nth\) usage phase is defined as

\[
pmf = (1 - p)^n \quad \{n=1, 2, 3, \ldots \ldots \}, \quad (n>0)
\]

The total number of times the device was reused, including the first usage stage, was calculated by deriving the expectation value \(E(X_n)\), as defined in (2), where \(X\) is a random variable with geometric \((p)\) distribution between 1 and \(n\):

\[
E(X_n) = \sum_{n=1}^{\infty} (1 - p)^n p \cdot n
\]

A notable challenge in reusing these smart mobile devices is that some of the hardware components wear out over long-term usage of the device. Typical problems that may be encountered due to long-term usage of these devices are damage to the touch screen display, malfunctioning of or damage to the touch screen sensors, end of battery lifetime, etc. Thus, it is assumed that in order to reuse the smart mobile devices and consequently execute any task under the thin-client paradigm, the device must have the following hardware components in a functioning state: LCD touch screen, Wi-Fi module, mother board, and battery. However, other hardware components such as cameras, GPS, and audio are considered task-dependent on the device being reused.

Considering the degradation of hardware components due to reusing the device, an energy cost to upgrade the hardware required to replace any components \((z_1e_m)\) was also included. Here, \(z_i\) is the fraction of manufacturing energy required for replacement of hardware components and varies between 0 and 1. Thus, the LCE consumption at each usage phase \((e_i)\) is the sum of energy consumed at the \(ith\) usage stage \((e_{ui})\) and the hardware upgrade energy cost \((z_{i-1}e_m)\) involved at that usage phase.
stage. The energy consumption at the $i$th usage phase is defined as

$$e_i = e_{ui} + z_{i-1} \cdot e_m \quad \{i=1, 2, 3, \ldots, n\}, \quad z_0 = 0$$ \hspace{1cm} (3)$$

Therefore, the total energy consumed by the mobile device throughout its life cycle includes the manufacturing energy, energy cost of each usage phase and hardware-upgrade cost involved each time the device is reused, and recycling energy:

$$e_{total} = e_m + \sum_{i=1}^{\text{ceil}[E(X)]} (e_i) + e_r$$ \hspace{1cm} (4)$$

Using (3), the average LCE of the device is calculated for the derived expected value of usage stages as

$$LCE_{avg} = \frac{1}{\text{ceil}[E(X)]} [e_{total}]$$ \hspace{1cm} (5)$$

Thus, the total energy consumed throughout the life cycle of the mobile device can be stated as in equations (1)–(4).

B. Numerical Analysis of Model

A numerical analysis based on the presented model was performed on smartphones using equations (3) and (5). The data for analysis was taken from Nokia’s technical report in [20]. The total life cycle energy of smartphones was evaluated for various selected geometric distributions of $p$. The $E(X)$ allows for determining the total number of usage stages a device traverses before exiting to the recycling phase with probability $p$. Figure 2 depicts the variation of total number of usage phases with the average LCE. The total LCE of the device includes the manufacturing energy, total usage energy cost (including multiple usage and hardware-upgrade cost), and recycling energy cost, as shown in (4).

![Figure 2. Average LCE vs. number of usage years](image)

The average LCE of the device was calculated using (5) for the derived expected value ($n$) of usage stages. It was assumed that each usage stage consists of 12 months. It is estimated that the commercial cost to replace components such as an LCD touch screen or battery is nearly equal to 5–20% of the cost of the new device. Therefore, our numerical evaluation includes a hardware-upgrade energy cost after each usage phase, as shown in (3). An upgrade cost of 5% was added at the end of each usage stage in those cases when the device enters another usage stage considering the hardware degradation due to multiple usage of the device.
Figure 2 shows the average LCE consumption for various \( p \) probabilities to exit to the recycling phase. As shown, the average LCE consumption decreases linearly as the usage phase increases from 1.5 years to nearly four years. Also, devices with a lower probability \( p \) to exit to the recycling phase undergo a greater number of usage phases. An approximately 11% reduction in the average LCE was observed as a result of three-months increase in the usage phase from 1.5 to 1.8 years. This is due to the fact that with a shorter usage stage duration, there exists a disproportion between the manufacturing energy and usage stage energy consumption. Thus, more sustainable devices can help achieve proper balance between the manufacturing energy and usage stage energy.

Recalling (5), it can be noted that the average LCE is directly proportional to the total energy consumed throughout the device life cycle. It was also observed from the analysis that increasing the number of usage stages and reducing the usage phase energy would effectively result in a reduction in the average LCE. Next, we consider the thin-client paradigm for reusing mobile devices, which is a cloud-based software approach used to increase the number of usage phases by providing the required upgrades through the cloud. Also, we investigated if older smartphones could provide a comparatively similar performance to that of new devices. The evaluation of performance and energy savings if reusing the device is crucial in order to comprehend the effects on quality of experience.

Section IV defines the analytical model for energy consumption and performance. Sections V, VI, and VII present the experimental methodology and empirical analysis of the proposed model.

IV. THIN-CLIENT APPROACH AND METRICS FOR MOBILE DEVICES

In this section, we describe the thin-client approach and analytical metrics for evaluation of performance improvement and energy savings. In these experiments, mobile devices were reused under the thin-client paradigm. The metrics defined in this section will be used in later sections to empirically evaluate the performance and energy consumption of thin-client mobile devices. The proposed analytical metric defines the theoretical conditions for the use-phase energy savings under the thin-client paradigm. We also define the performance improvement of old mobile devices and end-to-end computation time to execute any task under the thin-client paradigm.

A. Thin-Client Approach of Mobile Computing

Thin clients are simplified devices with conceivably fewer hardware components compared to traditional mobile devices. The capabilities of mobile devices could be greatly enhanced when used with relatively high-performance server clouds over a network. In the thin-client paradigm, most of the computation tasks associated with applications can be offloaded to a remote cloud server, as shown in Figure 3. The thin client or (simply client) displays only the graphical output. The offloading approach
also improves the user experience by increasing the execution speed of the tasks via more capable remote servers. A step in this
direction is *ChromeBooks* by Google, where there is a limited operating system locally and most of the functionality is provided
by remote cloud servers.

A thin-client-server paradigm for smartphones can be effectively implemented for reutilization of old smartphones, thereby
reducing the overall energy consumption and e-waste from these devices. In this approach, smartphones must have the active
network interface in order to connect to the remote server. To execute any application in the thin-client-server approach, the
client device is required to set up a session with the remote server using a transmission control protocol/Internet protocol
(TCP/IP) socket interface.

A thin-client-server paradigm for smartphones can be effectively implemented for reutilization of old smartphones, thereby
reducing the overall energy consumption and e-waste from these devices. In this approach, smartphones must have the active
network interface in order to connect to the remote server. To execute any application in the thin-client-server approach, the
client device is required to set up a session with the remote server using a transmission control protocol/Internet protocol
(TCP/IP) socket interface.

Figure 3. Thin-client approach for mobile devices

The socket interface attaches the client application processes to the remote server. Data is sent and received by the client to
the remote server through the socket. To implement the client-server TCP/IP socket at the application level for the thin-client
approach, first, the socket is created on the client. Next, a transport address or port number is assigned to the socket. Then the
client device sets up a connection with the remote server on the specified port number. Before the client connects to the server,
the remote server must have a socket ready to accept the client connection. Once the server accepts the socket, the client and
server can communicate to send and receive any data over the network. The next section explains implementation of the mobile
thin-client approach for reutilization of old smartphones.

### B. Conditions for Energy Savings under Thin-Client-Server Model

The conditions essential for saving energy on a mobile device by means of the thin-client paradigm are discussed in this
section. The following assumptions were considered in order to justifiably evaluate the conditions for energy savings:

i. For the computations to be performed locally on the device, the communication interface does not need to be active.

ii. To ensure that applications are executed locally, they can be pre-installed on the mobile device. In such a case, there is
   no communication between the mobile client and the remote server.
iii. In the case of local execution, all computations required by the application will be executed by the processor on the local device.

iv. In the case of remote computations, or the thin-client-server paradigm, access to the application is provided through remote cloud servers using the device’s Wi-Fi interface.

v. It is assumed that the base power and power consumed by the display remain constant for both cases, i.e., when the application is executed locally versus execution of the same application remotely under the thin-client paradigm. Therefore, a necessary condition for an execution to be energy efficient under the thin-client paradigm is that the energy consumed for remote execution ($E_R$) is less than the local execution energy ($E_L$), as defined by

$$E_R < E_L$$  \hspace{1cm} (6)

where $E_L$ could be defined as the product of the average power consumed by the mobile client CPU ($P_L$) and the time ($t_L$) required to process the task, as shown in (7). An additional assumption in this scenario is that the Wi-Fi interface is considered to remain idle because the local execution does not require any communication over the wireless network.

$$E_L = P_L \times t_L$$  \hspace{1cm} (7)

Similar to $E_L$, the energy consumption of the mobile device when the task is executed remotely ($E_R$) under the thin-client paradigm could be defined as the product of the average power consumed by both the network interface and the mobile client CPU ($P_R$) and the total end-to-end time for remote processing ($t_R$):

$$E_R = P_R \times t_R$$  \hspace{1cm} (8)

The total end-to-end time for remote execution includes time consumed by the client device to send and receive data from a remote server ($t^{NIC}_R$) and the time required by the remote server to complete the task ($t^{CPU}_R$):

$$t_R = t^{NIC}_R + t^{CPU}_R$$  \hspace{1cm} (9)

The completion of the application execution also includes the time required to offload the data entirely to the remote server for processing and for network latency. The time required for such offloading could be determined to be twice the one-way network latency (OTT) between the client and the server:

$$t^{NIC}_R = 2 \times OTT$$  \hspace{1cm} (10)

In this scenario of remote execution via the thin-client-server model, it was assumed that the remote server execution time is $n$ times faster than that of the local device. Hence, the performance improvement under thin-client paradigm could be defined using the term Speedup. Speedup can be defined as the ratio of the time required for local execution ($t_L$) to that of the time required for remote processing ($t_R$):
The overall percentage energy savings in the thin-client-server scenario is dependent on energy consumption of both the local and remote execution:

\[
Speedup = \frac{t_s}{t_r} = n
\]  

\[
E_{savings\%} = \frac{E_L - E_R}{E_L} \times 100
\]

V. EXPERIMENTAL METHODOLOGY

The proposed thin-client approach explained in Section IV was experimentally implemented by means of a laboratory-based setup of the thin-client scenario. An assortment of tests was performed in order to evaluate the accuracy and applicability of the proposed approach. The goal of our experimental study was to compare and analyze the performance and energy consumption of reused mobile devices operating under thin-client paradigm with that of newer mobile devices with superior hardware capabilities.

A. Setup for Laboratory Test-Bed

A well-known limitation for testing the proposed thin-client-server architecture with the prevailing applications is the lack of open-source applications. In order to overcome this challenge, we developed an Android-based application similar to that of a typical word-search application. This application was used to perform word searches for a preselected word or phrase in the input word file. The application was also equipped with a counter to keep track of the number of appearances of the selected word or phrase in the entire input file. The application uses a regular expression-matching pattern to locate the searched set of words in the input file. Input files consisting of different sample sizes (total number of words) were considered in order to improve the accuracy of the test results. The test word files were stored in an asset folder separately on the local device as well as the remote server for access during application logic executions. Eclipse IDE was used for the Android programming to facilitate automatic rebuilding of the project each time changes were made to the source code [21]. The Android Development Tool (ADT) plug-in was installed on the Android 2.2 platform for implementation of the algorithm and development of the test applications. The apk file was installed on multiple client devices to execute and test the applications [22]. The client mobile devices used for the empirical testing are listed in Table II. Devices such as the Motorola MB865, HTC One V, and HTC desire, with approximate usage ages of 1.5, 1, and 3.5 years, respectively, were chosen as test devices. Motorola’s MB865 is the higher-end device, with nearly double the size of random access memory (RAM) and about 0.2 GHz higher CPU clock frequency, compared to the other two devices.

<table>
<thead>
<tr>
<th>Model</th>
<th>Device Age (years)</th>
<th>Android Version</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>CPU Frequency</td>
</tr>
</tbody>
</table>

Table II. Smartphones used for empirical study
Motorola MB865   1.5  4.0.4 Dual-Core 1.2 GHz Cortex-A9  1 GB
HTC One V 1 4.0.3 1 GHz  512 MB
HTC Desire 3.5 2.2 1 GHz Scorpion  576 MB

In order to provide a methodical comparison of power consumption for local computation-based and thin-client paradigm-based executions, the same applications were implemented in each test measurement after installing them on the local device as well as at a remote location on the server. Experiments consisted of search operations performed to find the word “Android” in text files with varying sample sizes. Table III presents the test cases, text file sizes, and number of appearances of the preselected word/phrase in the text files for each case. During the experiments, all sync services on the device were disabled in order to isolate the current operation. The Wi-Fi and 3G interfaces were turned off while executing the applications locally. The Wi-Fi interface was the only external access application that was enabled to run while executing the search operation remotely.

Table III. Test cases evaluated for empirical study

<table>
<thead>
<tr>
<th>Test Case</th>
<th>File Size (KB)</th>
<th>Number of Appearances of Preselected Word/Phrase</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>56</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>112</td>
</tr>
<tr>
<td>3</td>
<td>25</td>
<td>140</td>
</tr>
<tr>
<td>4</td>
<td>75</td>
<td>420</td>
</tr>
<tr>
<td>5</td>
<td>125</td>
<td>700</td>
</tr>
<tr>
<td>6</td>
<td>250</td>
<td>1,400</td>
</tr>
<tr>
<td>7</td>
<td>650</td>
<td>3,500</td>
</tr>
<tr>
<td>8</td>
<td>1,016</td>
<td>5,822</td>
</tr>
</tbody>
</table>

B. Framework for Application Execution Locally on Device

The application logic for traditional or local execution of the applications was written for implementation on the client device, i.e., the smartphone. As a typical characteristic of all smartphones, localized programs utilize the CPU and other resources of the device for computations. A simple graphical user interface (GUI) consisting of a button labeled LocalSearch was developed via Android programming, as shown in Figure 4(a). Each click of this button generates an event followed by the extraction of the input file in the asset folder. The name of the file in which the word/phrase to be searched was entered via the GUI. The word-searching operation was performed on the entire document using a regular expression, and the count was maintained using a for loop that traverses the input file line by line.
C. Framework for Remote Execution of Application via Thin-Client Scenario

In the thin-client application framework, the program is initiated on the client device, i.e., the smartphone, but the computation is executed on the remote server. The remote execution of the client and server scenario was implemented using socket programming, as shown in Figure 5. The Android smartphone was installed to act as the client. In this experiment, the remote server was a laptop equipped with a Core i5 @ 2.3 GHz 64-bit processor, 4 GB of RAM, and a 500 GB hard disk drive, and the client was an Android smartphone with different specifications, as shown previously in Table II.

Similar to the setup for the local execution scenario, a simple GUI consisting of a button labeled RemoteSearch was developed, as shown in Figure 4(b). The GUI included a field to enter the name of the file being tested and the IP address of the remote server. With an On Click of the button, an event is generated, whereby the input file in the asset folder is extracted to retrieve the values to be computed. Then the input file extracted from the asset folder is sent to the remote server as a byte stream.
to the ObjectInputStream. After the client requested task is completed, the server sends the results back to the client. The final result is displayed on the client device (smartphone). In order to reduce network delays, the server and client were designed to reside in the same IP subnet. All experiments were performed under the same network conditions.

VI. EVALUATION OF PERFORMANCE IMPROVEMENT

This section presents the empirical analysis of performance improvement in terms of Speedup, as defined previously in Section IV for reusing a mobile device under the thin-client paradigm. First, the Speedup measurement methodology is presented followed by results and analysis.

A. Measurement of Speedup in Local vs Thin-Client Execution

Android’s Traceview tool was used to measure the end-to-end time expended by the test application. This tool generates trace files with a graphical presentation along with the detailed CPU time lapse analysis after execution of the application. In the Time panel of the tool, each method from the program is represented by a different color code. The method executions in the program are shown in chronological order, with time represented in milliseconds. The profile panel in the tool displays all synopses of time being spent in the method. It also displays inclusive time (time spent by the parent method and time spent by the child method in milliseconds) and exclusive time (time spent by the method alone in milliseconds). Because the inclusive CPU time comprises both time spent in the parent method and time spent in any child method (referred to as function), inclusive CPU time was the time factor parameter considered for this experiment. To measure the CPU time consumption, tracing was performed from the first instance of input data file extraction until the result is extracted back by the application program.

In the case of a local computational program, time was logged from the click of the LocalSearch button (event generation for computation) until the result was generated. For the remote computational program, this log includes the time tracked from the instant click of the RemoteSearch button (event generation for computation), the time taken for sending the file as an Object to the remote server (laptop), until the result was generated and sent back to the calling program on the client (smartphone). This remote execution time also includes the time to send and receive the file to the remote server, time required by the remote server to process the task, and the network delays present in the network. After the completion of the application execution, the trace log file “.trace” was generated and saved with the specified name in the storage device (SD) card of the Android phone.

B. Speedup Comparison of Local and Remote Execution Scenarios

The plots shown in Figures 6 and 7 indicate the computation time for execution of the word-search operation versus file size for local-based and thin-client-based scenarios, respectively.
The time required for remote executions is the total time \( t_E \), which includes the time for remote processing, sending, and receiving the final result. It can be clearly seen from Figure 6 that in the case of local executions (on the smartphone) of the application, the total CPU time for computations increases exponentially with increasing file size. It can also be seen in Figure 6 that the computation time required for the relatively older devices, such as HTC One V and HTC Desire, increased approximately 11 times as the file size grew from 10 KB to 1,016 KB. However, for the same variation in file size, the computation time for the Motorola MB865 increased approximately eight times, which is evidenced by its higher-end CPU and RAM capabilities. A significant result is the nearly 30% lower computational time required by the comparatively new Motorola MB865 to process the larger files in the range of 650 KB and ~1 MB, compared to the other two devices with higher usage lifetimes. Hence, the Motorola MB865 is comparatively dominant in performance for the case of local execution of the application as a result of its superior hardware capabilities.

Under the thin-client paradigm (remote execution), the CPU utilization time increased almost linearly with an increase in the file size, as depicted by Figure 7. The maximum computation time in each test case for all devices was significantly lower in the
case of remote execution, irrespective of file size. Figures 8(a) to 8(c) depict the magnitude of reduction in computation time via remote processing of the application for the HTC Desire, HTC One V, and Motorola MB865, respectively. Such a low CPU time utilization in the thin-client scenario is a direct consequence of comparatively fewer computations executed by the local smartphone. In this case, the CPU is only utilized for sending input data to the remote server and for subsequently displaying the final results.

Another notable observation from Figures 6 and 7 is that even though the newer Motorola MB865 possesses superior processing capabilities, it requires about 0.82 ms for processing a ~1 MB file. However, the computation time required to process the same file via the comparatively older smartphones (HTC Desire and HTC One V) through remote execution is 0.32 ms and 0.34 ms, respectively. This computation time for the remote execution via smartphone with higher usage lifetimes was lower by nearly 60%. Such a beneficially high reduction in the computation time shows that the devices with higher usage lifetime could result in efficient performance with the help of the thin-client server paradigm. Such benefits essentially reduce the probability of discarding used smartphones and, in turn, help in lowering the e-waste by enhancing the reutilization capabilities of these devices. Results also indicate that the reutilization of these devices would not significantly affect the end-user experience because most of the resources in the local device will only be used to display the results of the remote computations.

Figure 8. Computation time vs file size for remote and local execution scenarios: (a) HTC Desire, (b) HTC One V, and (c) Motorola MB865
The beneficial reduction in processing time via the two older devices (HTC Desire and HTC One V) while remotely processing the files under the thin-client paradigm was found to be particularly significant for larger file sizes, as shown in Figure 9. As can be seen, for processing larger files, the reutilization of the chosen older devices under the thin-client paradigm results in lower computation times, compared to performing the same tasks locally on newer devices equipped with better hardware capabilities.

As can be observed in Figure 9, over a file size range of 10 KB to 75 KB, the computation time required for local processing on the Motorola MB865 is nearly 90% lower than that of remote processing via the HTC smart devices. However, the computation time required by the Motorola MB865 for processing larger files significantly escalates. Conversely, the older HTC smart devices (Desire and One V) require a remarkable 57% and 50% less computation time to process the 650 KB and ~1 MB files, respectively, via remote processing compared to local processing of the same files through the relatively newer Motorola MB865. The notable escalation in computation time above a file size of 250 KB arises from the fact that the number of words to be searched in the larger file size also increases proportionally (see Table III), which in turn increases the computation time and load on the processor. Moreover, it should also be noted that in the case of remote processing, the end-to-end time is dependent on the network latency and the processing capability of the remote server.

![Figure 9. Comparison of CPU computation time versus file size for remote processing on HTC (higher usage lifetime) and Motorola MB865 (lower usage lifetime) devices](image)

Furthermore, the efficiency of the thin-client paradigm was analyzed by calculating the Speedup defined in (11) based on the experimentally obtained computation times. Speedup effectively provides an estimate of the relative improvement in computation time under the thin-client paradigm. Figure 10 comprehensively depicts the calculated Speedup versus file size for each device tested. From the definition of Speedup in (11), it is expected that the lower the time for remote processing \( t_R \), the higher the Speedup. All three devices tested showed increasing Speedup with an increase in file size. However, it was observed that for file sizes equal to 125 KB or higher, Speedup for the two HTC devices (higher usage life time) was consistently higher than that of the Motorola MB865 (low usage lifetime). The dominance in Speedup of the HTC Desire was particularly higher by
25% and 65% for file sizes of 650 KB and ~1 MB, respectively. The inferior Speedup for larger file sizes in the case of the Motorola MB865 is a direct result of the low \( (t_R - t_L) \) compared to the HTC Desire, as shown earlier in Figures 8(c) and 8(a), respectively. It should also be noted that according to (9), \( t_R \) consists of both the time required to transfer the file for remote processing (network latency) and the OTT time as discussed in Section II. Hence, \( t_R \) varies for each device tested, even though the remote server hardware specifications remain unchanged.

![Graph showing Speedup under thin-client paradigm](image)

Figure 10. Calculated Speedup under thin-client paradigm (remote processing)

VII. EVALUATION OF ENERGY SAVINGS

This section presents the estimation of usage phase energy savings defined in Section IV for reusing mobile devices under a thin-client paradigm. The power measurement methodology is presented first, followed by results and analysis.

A. Power Measurement Methodology

Based on the experimentally obtained \( t_R \) and \( t_L \) for all devices tested, the percentage of energy savings was calculated using (12). The corresponding power consumptions in each case were measured using the Monsoon Power Meter, as shown in Figure 11, and a commercially available power measurement application, namely PowerTutor [23, 24, 25]. A subtractive method was used to log power using the Monsoon Power Meter. Initially all the sync services and Wi-Fi and 3G interfaces were turned off, and the base power consumption was logged. Power was logged for each individual test case for both local and remote execution of the applications on multiple devices.
In order to calculate the total power in the local ($P_L$) and remote ($P_R$) execution scenarios, the base power was subtracted from the measured power consumption during execution of the application, consistent with our assumption v in Section IV. The power consumption was also logged using the PowerTutor application for every millisecond, and each application was mapped with a unique user identification (UID) number [25]. An automated program was developed in Java to calculate the total power consumption ($P_L$ and $P_R$) using the power log from PowerTutor using equations (7) to (13).

B. Estimation of Energy Savings under Thin-Client Paradigm

Figure 12 shows the calculated percentage of energy savings versus file size for our test-bed setup under the thin-client paradigm. As shown, for all devices, this savings is negative for smaller file sizes, thus indicating that no energy could be saved for these test cases. An analysis of the power logs indicated that in this case, more energy was consumed by the wireless interface of the device while communicating with the remote server. However, in our test setup, the energy savings retains a positive value for file sizes above 75 KB, thus showing that energy could be saved only on mobile devices under the thin-client scenario if the processing energy consumed on the local device is greater than the energy consumed by the wireless interface to process and send the data for remote processing. It is also shown in Figure 12 that for the test case with ~1 MB file size, approximately 85% of the energy could be saved under the thin-client-server paradigm for all devices. Such a high percentage of energy savings was achieved by offloading a major fraction of the computations to the remote server. However, as per our test setup, there exists a trade-off between wireless communication for remote processing and local computations for smaller file sizes. This suggests that for both newer and older devices, if the application requires less computation, then it is energy efficient to handle such a task locally. Nevertheless, it is always beneficial for larger and computation-intensive files to be processed under the thin-client scenario in order to lower the usage phase energy and achieve a longer usage lifetime. Energy savings could
also be affected by varying the network conditions, form factor of the devices, remote server capabilities, etc., as reported earlier in our previous work [11, 19].

![Percentage energy savings versus input file sizes for devices tested](image)

Figure 12. Percentage energy savings versus input file sizes for devices tested

VIII. CONCLUSION

In this work, we presented a thin-client approach for reutilization of smartphones. First, an analytical model was developed to understand the imbalance in the life cycle energy due to the short lifespan of smart mobile devices. A comprehensive numerical analysis was performed to calculate the average LCE reduction as the result of reutilization of discarded mobile phones. A reduction in the average LCE of approximately 11% was observed as a result of a three-month increase in the usage phase. It was observed that with a shorter usage duration, a disproportion exists between the manufacturing energy and the usage energy consumption. Thus, more sustainable devices can help achieve proper balance between the manufacturing energy and usage energy. Furthermore, the thin-client paradigm for reusing mobile devices was empirically tested.

The proposed thin-client approach for smartphone mobile devices was experimentally implemented by means of a laboratory-based setup of the thin-client scenario. The empirical study compared and analyzed the performance and energy savings of reused mobile devices operating under a thin-client paradigm with newer mobile devices having superior hardware capabilities. An Android-based application, similar to that of a typical word-search application, was developed to perform word searches for a pre-selected word or phrase in the input word file.

Future work could include additional case studies to analyze the feasibility for reusing mobile devices under the thin-client paradigm for various purposes such as classroom learning, research, and educational laboratories. The development of various educational applications and laboratory test-beds will be targeted to extend the performance study discussed in this paper.

REFERENCES


**SHORT BIOS**

**Toolika Ghose** is Senior Network Security Engineer at Brown University. She received her Ph.D. in Electrical Engineering from Wichita State University in 2013. Her research interests include green computing, sustainability, mobile computing, network security, and the Internet of Things.

**Vinod Namboodiri** is Director of the WiNeS laboratory and currently Associate Professor in the Department of Electrical Engineering and Computer Science at Wichita State University. His research interests are in designing medium access control routing and energy-efficient protocols and algorithms for different wireless technologies, designing communication techniques for smart electric grids, and integrating sustainability in mobile computing.

**Ravi Pendse** serves as the Vice President and Chief Information Officer at Brown University. He is a Brown University Cisco Fellow and a senior member of IEEE. He is also a Professor of Practice in Computer Science and Engineering. His research interests include the Internet of Things, adhoc networks, mobility, and security.