TNC Transactions on Networks and Communications
SOCIETY FOR SCIENCE AND EDUCATION UNITED KINGDOM
VOLUME 2, ISSUE 3
ISSN: 2054 - 7420

Cloud Computing and Offloading Framework for enhancing Android Smart Device Battery Life

Ahmed H. Najim, Shawkat K. Guirguis, Magda M. Madbouly
Department of Information Technology
Institute of Graduate Studies & Research, Alexandria University
Alexandria, Egypt
Ahmed_al_adhami@yahoo.com, Shawkat_g@yahoo.com, mmadbouly@hotmail.com

ABSTRACT

Information Technology (IT) field before cloud computing is not the same after. It has recently accelerated as new criteria for presenting and delivering services over the Internet. Cloud Computing is reacting positively as a concept that may greatly improve the smart devices (Smartphone, Tablet and Phablet) reliability. This paper mainly concentrates to enhance the smart devices dependability through and applications (app) during offloading of services among cloud and device to save energy, many challenges had faced that idea ex: unstable network, unstable Internet services, Synchronization and the limitations of open source smart devices operating system (OS). In order to cure the earlier challenges, An Android OS used with a new layer to offload the non-critical app's jobs. That should occur in One's of two clouds developed specially to do that, the first cloud is a private, connected with smart device through the same data network on-line and the second cloud is a public, related to the previous cloud and the smart device off-line when Internet exists. Using this system, Smart device users may choose to run their mobile application jobs either locally or offloading it to the cloud, this shall save energy spent from the smart device battery when using Third generation of mobile network (3G) or WI-FI. This system has achieved in the provision of high-energy reached 98.67% and will be seen in the context of this research paper.

Keywords: Offloading, Energy, Power consumption, Smart Device, Android, Mobile cloud computing.

1. INTRODUCTION

Second millennium has seen much technological progress, which was clearly highlighted in the mobile phone sector. Handheld mobile technology is reaching first responders, disaster-relief workers, and soldiers in the field to aid in various tasks, such as speech and image recognition, natural-language processing, decision making, and mission planning. [1] Mobile

DOI: 10.14738/tnc.23.236
Publication Date: 12th June 2014
URL: http://dx.doi.org/10.14738/tnc. 23.236

phones have been undergoing a breathtaking evolution over the last two decades, starting from simple devices with only voice services towards smart device offering novel services such as mobile Internet, high data rate connectivity and many more. Smart devices are battery driven to allow the highest degree of freedom for the user and the battery has to empower all the nice new features of a smartphone. Computation-Intensive tasks consume a large amount of power. At the same time, new expression appeared on the world of information technology called Cloud computing [2] a topic that received a great deal of attention from individuals and organizations from different disciplines in the last decade. This new environment implies a high flexibility and availability of computing resources at different levels of abstraction at a lower cost. The concept of Mobile Cloud Computing (MCC) intends to make the advantages of Cloud Computing available for mobile users, but will provide additional functionality to the cloud, as well. MCC will help to overcome limitations of mobile devices in particular of the processing power and data storage. It might also help to extend the battery life by moving the execution of commutation-intensive application "to the cloud." In this proposed system, developed mobile cloud computing model, provided the environment specifically designed for smart device users. This system allows users to create virtual smartphone images in the public cloud and remotely run their jobs in these images as they would locally By offloading only non-critical jobs when sustainable network existed to synchronization files. Both smart device and the private cloud hosting computer are connected to the same wireless local area network). Private cloud used as a bridge between the public cloud and the smart device. The main objectives of that paper are prolonging the battery life of smart devices by offloading non critical jobs of it with the cloud. Clarify how cloud computing services can increase the effectiveness, dependability, mobility and reliability of the android smart devices in the future. Figure 1 shows the basic proposed idea outline.

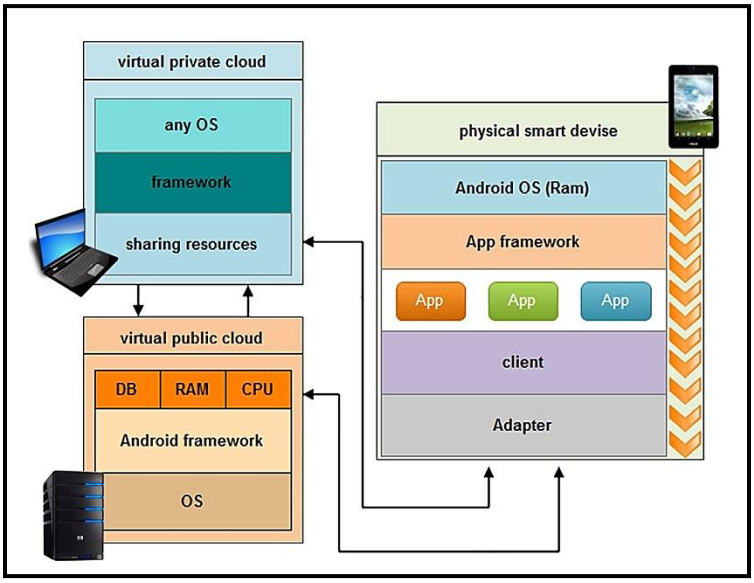

Figure 1: The basic idea Outline

2. RELATED WORK

Reducing power consumption in smart devices always had been a significant target for researchers and companies. After the huge revolution in the mobile sector and the release of a new category called smart phones characterized by the availability of using the internet, applications and many other features. Thus, all of those features are drying batteries faster than ever. Many papers, articles and theses written about that topic, each group of these research papers are trying to solve this problem in a different direction. Some of these papers are comparing smartphones hardware components power consumption through measuring and comparing an energy consuming entities such as wireless air interfaces, display, music player and others who trying to develop and modified the android java code. as mentioned in [1, 4, 10, 11, 12, 13, 14]. Others studies are tried to solve that paper through developing or designing an offloading framework and using other offloading techniques which are a promising way to improve and to reduce the battery power consumption of a smartphone application. That can be achieved by executing some parts of the application on a remote server decide at runtime [4, 5, 6, 7, 8, 9, 15]. The proposed system in this paper most closely related to Chun et al. [3], because of sharing some of the objectives and focusing on mobile applications. However focusing more on offloads the jobs Instead of the whole app.

3. PROPOSED METHODOLOGY

The proposed system comes with better power saving performance. it composed of one external smartphone client and two front-end server(private cloud and public cloud).The client application modified job list and Synchronizing that list with each cloud list in order to determine the non-critical files (jobs) which is want to be downloaded through that system. The following subsections discuss each step in details.

3.1 Proposed System Software Components

3.1.1 Client (Android App)

- The main function for that part of proposed system is to link the local smart device with clouds. That will to update (add or remove) downloading jobs and to choose either to download them using proposed offloading framework or by direct download through any available data network And receiving downloaded file through WI-FI connection when it connected at the same data network with private cloud.

- Technologies used to design that app are Java android programming language, Eclipse Android emulator and editor, Version: 3.7 Indigo Service Release two with Java runtime environment (JRE). Also, AVD Manager and Android SDK that provides Application programming interface (API) libraries and developer tools necessary to build, test, and debug apps for Android.

- As shown in Figure 2 this framework includes of several parts and layers. The first part is the App power consumption adapter and this part would determine the amount of activity applications and energy consumption as well as it works as a decision maker to determine the method of loading. The second part is smart device sharing resources which include (RAM, CPU and storage). The third part is synchronization layer which is obviously responsible about all synchronization operations at that framework. The fourth part is private cloud handler which is responsible for executing and managing connection form with private cloud. The fifth part is public cloud handler which is doing Formulation of the updated version of the job's list and governs the relationship between the public cloud and local smart device. The final part of that framework is the power consumption manager which working as an interface layer for proposed application framework to related private cloud and public cloud.

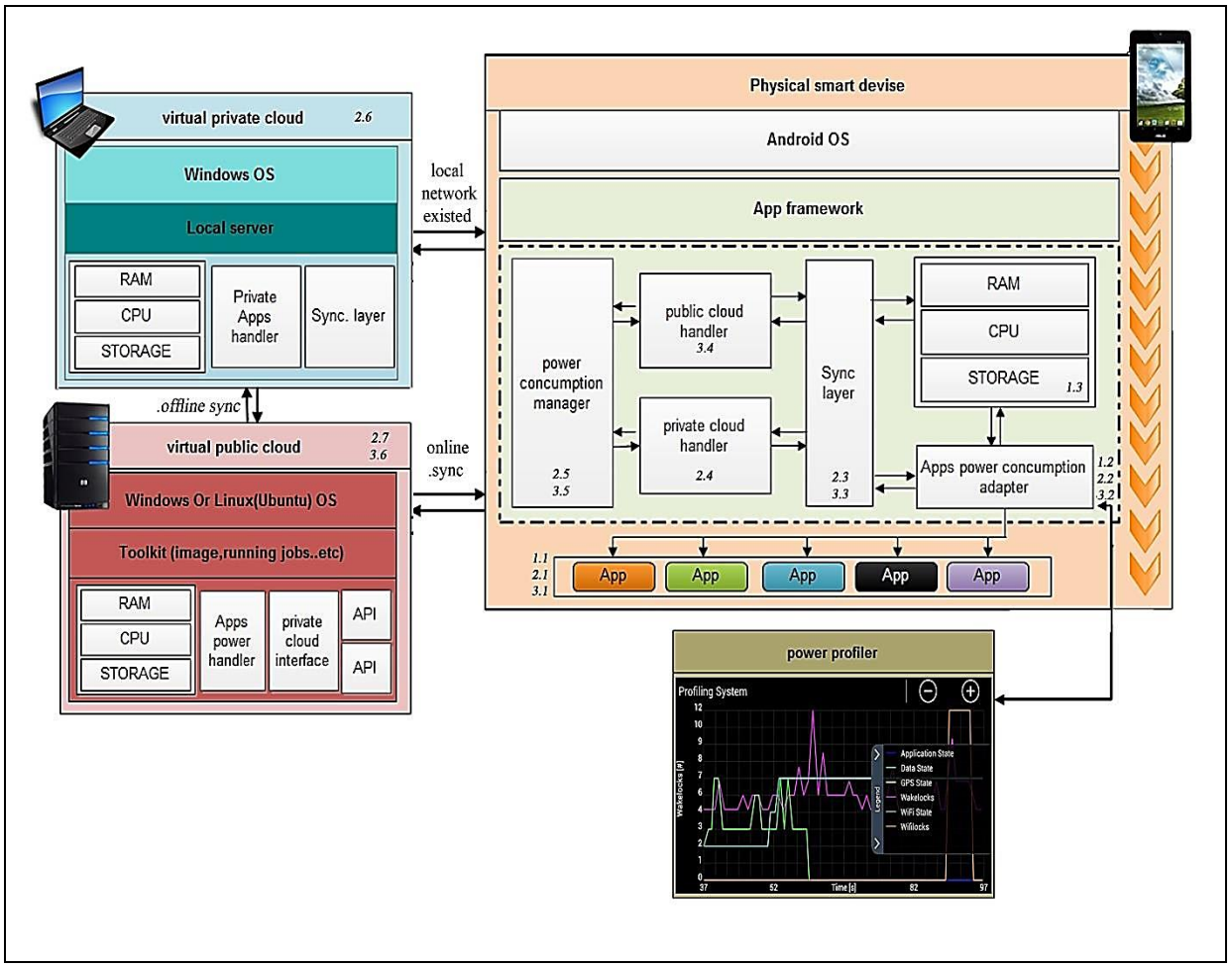

Figure 2: Proposed system prototype

3.1.2 Public cloud

- The main function for that part of proposed system is to creating, Operating and managing virtual image of the local smart device jobs. It is also responsible for updating and Synchronizing download jobs list and download selected jobs waiting to be sent to a private

cloud when internet exists. Then send it back to the client device as soon as it connected with private cloud at the same sustainable data network.

- Technologies used to design that cloud are PHP (server-side scripting language), MySQL (open-source relational database management system (RDBMS)), NetBeans (integrated development environment (IDE)), and XAMPP, Apache HTTP Server, Navicat, Symfony PHP web application framework and Doctrine (PHP) library.

- As shown in Figure 2 public cloud includes of several parts and layers. The first part is windows OS layer which operating server. The second part is a toolkit which is responsible for managing virtual image. The third part is server sharing resources which include (RAM, CPU and storage). The fourth part is Apps power handler which is responsible for managing apps downloading jobs and supervising its progress. The fifth part is private cloud interface which is responsible for managing the connection with private cloud and updating the job's list. The final part of that framework is APIs that governs the form that public cloud components should interact with each other.

3.1.3 Private cloud

- The main function for that part of proposed system is updating and Synchronizing download jobs list and received download files from public cloud and saving it as temporary files. Then send it back to the client device as soon as it connected with it at the same sustainable data network.

- Technologies used to design that cloud are JavaScript programing language, CSS, NetBeans (integrated development environment (IDE)), and Notepad++ source code editor.

- As shown in Figure 2 private cloud includes of several parts and layers. The first part is windows OS layer which operating server on the local server on laptop or desktop computer. The second part is a local server. The third part is computer sharing resources which include (RAM, CPU and storage). The fourth part is a private Apps handler which is responsible for managing power consume for android app image over downloading. The fifth part is private cloud interface which is responsible for managing the connection with private cloud and updating the job's list. The final part of that framework is APIs (Application programming interface) that governs the form that public cloud components should interact with each other.

3.2 Proposed System Hardware Components

The framework, presented in this paper implemented with three devices. The first one is a laptop computer with the following specifications: Intel® Pentium® processor B950 (2.1 GHz, 2 MB) with 6GB RAM DDR3, 1333 MHz, Video Graphics Intel® HD Graphics3000, Video Memory up to 1695 MB and Integrated WLAN Intel® Centrino® Wireless N1030 (IEEE 802.11b/g/n). This machine equipped with operating system Windows® 7 Ultimate 64-bit service back 1. The second device which been used is an Android smartphone with the following specifications: is

equipped with operating system Android OS, v2.3.6 (Gingerbread), CPU: Dual-core 1 GHz Cortex-A9, internal memory 8/16 GB storage, 768 MB RAM, 2 GB ROM, Li-Ion 1500 Milli Amp Hour (mAh) battery, compatible with 3G cellular network: HSDPA 850 / 900 / 1900 / 2100,Wi-Fi 802.11 a/b/g/n, dual-band, DLNA, Wi-Fi hotspot. The third device which been used is a 150Mbps Wireless N ADSL2+ Modem Router with the following specifications:4 10/100Mbps RJ45 Ports, 1 RJ11 Port, This router is equipped with Wireless Standards IEEE 802.11g, 802.11b, with some 802.11n features and Frequency run to 2.400-2.4835GHz.

3.3 proposed system working scenarios

This system exploiting one of the functions that are performed daily through smart devices namely download function. This function existed in any desktop computer, laptop computer and Smart device. Figure 2: shows the prototype of the proposed system, describing the system main components (client device, private cloud and public cloud). It determines three possible scenarios for downloading file as in below.

3.3.1 The first scenario

This scenario assumes that there is available sustainable wireless network (WI-FI) connecting both client device and private cloud and internet service. The first scenario starts with apps that need a file to be downloaded. The Apps send a request to App power consumption adapter which is connected to power profiler .this adapter is working as decision layer decide which file is downloaded locally and which one is downloaded through proposed framework and clouds automatically or manually as done in proposed prototype. Then moving to the synchronization layer, which is responsible for all synchronization operations (job list, job status). Then moving to private cloud handler, which is responsible for executing and managing connection form with private cloud. It's also formulation of the updated version of the list of jobs. Finally moves to the power consumption manager which working as an interface layer for proposed application framework to related private cloud which is connecting offline with public cloud.

3.3.2 The second scenario

This scenario assumes that there is no sustainable wireless network (WI-FI) connecting both client device and private cloud. However, there is an internet service. This scenario starts with the same previous steps until reaching the synchronization layer, which is responsible for all synchronization operations (job list, job status). Then moving to public cloud handler to do Formulation of the updated version of the job's list, in order to post it finally to the power consumption manager and send selected jobs to the public cloud directly. Public cloud should download those jobs to the private cloud and waiting for sustainable connection between the last and client device to finally deliver those files.

3.3.3 The third scenario

This scenario assumes that there is no sustainable wireless network (WI-FI) connecting both client device and private cloud. However, there is an internet service. It starts with apps that need a file to be downloaded. The Apps sends a request to App power consumption adapter which is connected to power profiler .this adapter decide automatically or manually downloading those jobs locally using available data network using device local resource.

3.4 Proposed system implementation steps

Figure 3 shows the general flow diagram of the implementation steps for proposed system, which comes with better classification performance. The first step in the process is running all system associated parts and then moving to the second step which should be include the installation process of the proposed Android application in the smart device and make sure it is ready for action. The second step should create an account on a public cloud plus create and install a virtual image of a smart local running Android. The fourth step process of installing a private Cloud on Desktop or laptop computer which is available as an infrastructure for this system. The fifth step in the process is creating a job list and synchronized between the client application in smart device, private cloud and public cloud. In the sixth step is the addition of a new job by adding job name and address (Uniform resource locator URL). In the seventh step, it highlights the fact that abstracts three main types of files that need to be downloaded frequently in smart devices as follows (operating system update file for android device client, android application update file and other files as (video files, audio files, other files). In the eighth step, a list of jobs will show the location of each file through the expression will be appeared in front of each file describing file (job) status. If the status is (**pending**), it means that the file is still in the phase of downloading between from public to private clouds. If the status is (**ready**), then it means that the file is already downloaded and exists temporarily in a private cloud waiting for a stable connection with the (client smart device) within the same data network in order to be sent. If the status is (**done**) then file downloaded successfully to its final destination (smart device), In step nine, an initialization the power profiler procedure starts to begin calculating the energy spent in the state of start downloading process and this depends on the question in step ten which Relation to the new files who show whether file is critical or not. The eleventh step will choose manually the way of downloading the file. After choosing the download process type flow will move to step twelve and that is where the stopping power profiler and followed step thirteen, which will report the energy consumption and there is the end of the process in step fourteen.

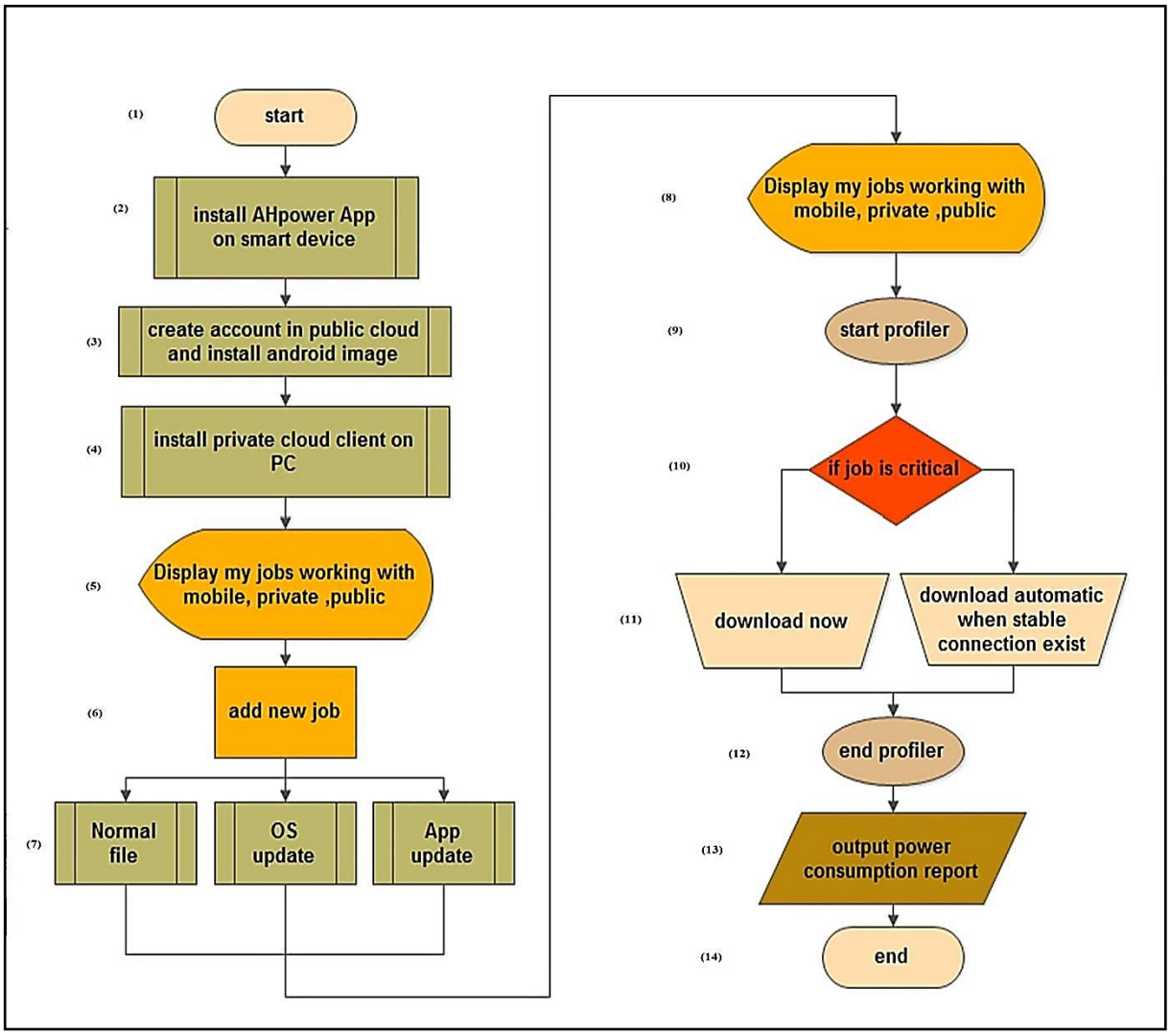

Figure 3: Proposed system implementation steps flow

4. EXPERIMENTAL RESULTS

4.1 Research Material

4.1.1 Used files

The proposed offloading framework (AHpower) described in chapter 3 has been tested using compressed Download Test Files from websites like http://www.thinkbroadband.com/download.html and other different types of files ex: (Audio, Video, RAR, Zip, pdf…etc.) also downloaded from web to compute downloading time and rate. The final database contains 40 downloaded files, each having different size between 1.25MB and 402MB. Files are downloaded in three downloading experiments,' each downloading experiment tested all 40 files with different experimental conditions as shown below in Table 1.

Table 1: Experimental used files

Sizes of downloaded Files used in each experiment in MB			
1.25	28.34	66.00	112.21
4.22	30.30	70.32	120.30
8.12	37.20	75.12	122.15
11.30	39.21	76.59	133.55
13.55	40.00	80.12	139.59
17.40	43.41	85.50	140.00
19.33	46.11	89.13	200.00
22.00	55.20	92.51	275.50
23.22	59.50	95.05	300.00
25.11	64.32	100.21	402.00

4.1.2 Used Devices

The framework, presented in this paper implemented with the same hardware components mentioned previously in 3.2**.**

4.1.3 Used Software

The proposed AHpower framework consist of three important components (client application (App), virtual private cloud and virtual public cloud) client App has been developed using several programming editors and emulators (Eclipse Android emulator and editor, software development kit (SDK)...etc.). Private cloud has been developed using (JavaScript language, CSS, NetBeans...etc.). Public cloud has been developed using (PHP, MySQL, Netbeans, Apache).All those software's Integrated together because of the high performance of its toolboxes And produce the AHpower prototype application compatible with android smart devices which are tested in this chapter.

4.1.4 Evaluation metrics

- **Objective:** the main objective of this work is to find a new idea to reduce battery power consumption in smart devices that are using Android as operating system.
- **Performance Measure:** the main scale Adopted to measure success or Failure of this work is Time spent in the job of downloading a file Thus the Spent energy during that time by PowerTutor profiler Which we will talk about in Comparison methods 4.1.5
- **Target:** creating offloading framework that or developing existed one to reach our objectives.

4.1.5 Comparison methods

In this chapter, an Android power profiler App measuring the power consumption for the smart device in the same time when downloading job occurred and gave the average of the consumed power for five minutes in Milli Watt (mW). Then demonstrate the effectiveness of the employed framework. PowerTutor profiler developed by the University of Michigan Ph.D. students. It allows seeing the impact of design changes on power efficiency, determining how actions are impacting battery life and monitoring the power consumption of any application. PowerTutor uses a power consumption model built by direct measurements during careful control of device power management states. This model provides power consumption estimates within 5% of actual values.

4.2 Experimental Results

In this section, an application to the power profiling is investigated to demonstrate the effectiveness of the employed system. In our experiments, we tested the whole files as a download jobs into three experiments. First experiment is using the proposed framework (AHpower) to execute downloading jobs. Second experiment is using the WI-FI to execute the same downloading jobs. Third experiment is using 3G cellular network to execute downloading jobs In order to make full use of the experiments and to evaluate the amount of power consumed in each experiment more accurately, then made a final comparison to prove our search point by results.

4.2.1 First experiment: download jobs using AHpower framework.

The first experiment was performed using different files to be downloaded for testing. By connecting the smartphone and running a private cloud on the laptop on the same local area network (LAN), a Synchronization start coordination of events to operate the system in unison, URLs of the files as downloading jobs, the results show a high performance. In contrast, the proposed system can be valuable when a huge amount of download jobs existed, which is important for classification make the suggested system outperforms other methods. Figure 4 shows the first experiment results using AHpower framework with WI-FI connection clarified that the average of total energy consumption in the smart device for five minutes = 1375 mW. That result should divide by 300 seconds as shown in equation (1) to get the energy consumption of every second, which is very critical to get accurate final results shown Table (2).

$$\text{Energy via WI-FI}=\frac{\text{Energy consumption value for N of minutes}}{\text{N of minutes}*60\ seconds} \quad (1)$$

Energy via WI-FI= 1375/5=275mW/minute, 1375mW/300sec =4.8 mW/sec.

Table 2: sample of downloaded files via proposed framework

Downloaded Job (file)	Via AHpower framework (APP)		
Size in (MB)	Time H,M,S,MS	Speed average Mb/s	Energy mW
1.25	00.00.00.11	100	0.52
4.22	00.00.00.36	100	1.72
8.12	00.00.00.67	100	3.21
11.30	00.00.00.95	100	4.56
13.55	00.00.01.13	100	5.42
17.40	00.00.01.46	100	7.01
19.33	00.00.01.62	100	7.77
22.00	00.00.01.85	100	8.88
23.22	00.00.01.96	100	9.41
25.11	00.00.02.10	100	10.08

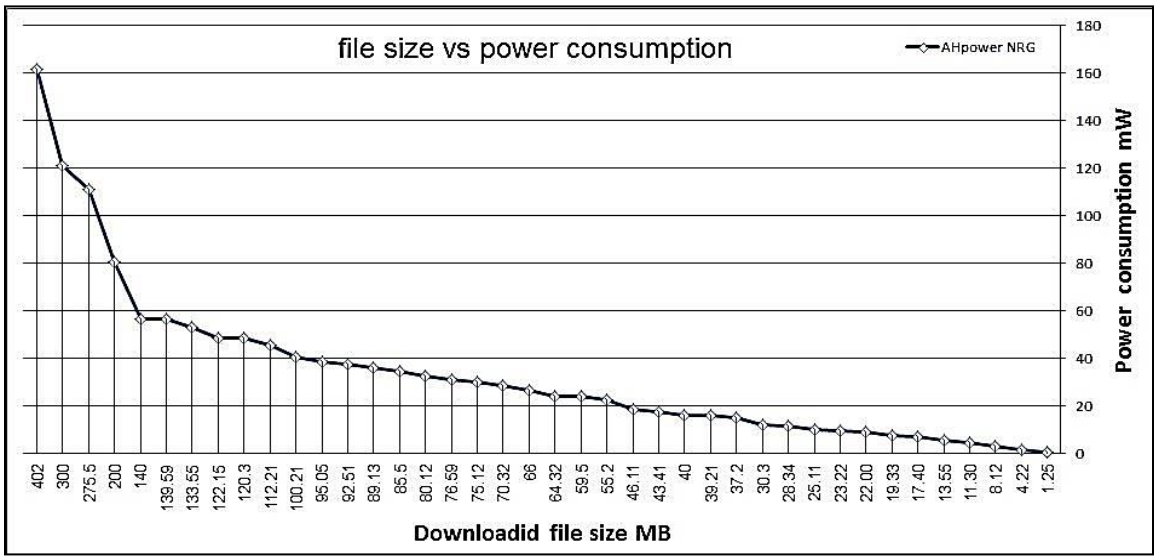

Figure 4: Experimental result using AHpower framework

4.2.2 Second experiment downloads jobs Wi-Fi network without AHpower

The second experiment was performed using the same files had been used in the first experiment to be downloaded for testing. Downloading the jobs take the exact first experiment steps but when putting the URLs but using the bottom (download without AHpower) when still connected to the internet through Wi-Fi. The results show the exact average of power but with a huge different in the downloading time (DLT). Figure 5 shows the second experiment results without using AHpower app with WI-FI connection clarified that the average of total energy

consumption in the smart device for five minutes matching the first result. As shown in equation (1), which is very critical to get accurate final results shown Table (3).

Table 3: sample of downloaded files via WI-FI

Downloaded Job (file)	Via (WI-FI)		
Size in (MB)	Time H,M,S,MS	Speed average Mb/s	Energy mW
1.25	00.00.05.11	2	24.48
4.22	00.00.17.27	2	82.89
8.12	00.00.33.26	2	156.32
11.30	00.00.46.28	2	222.14
13.55	00.00.55.48	2	266.30
17.40	00.01.11.27	2	342.10
19.33	00.01.19.18	2	380.06
22.00	00.01.30.14	2	432.67
23.22	00.01.35.11	2	456.53
25.11	00.01.42.85	2	493.68

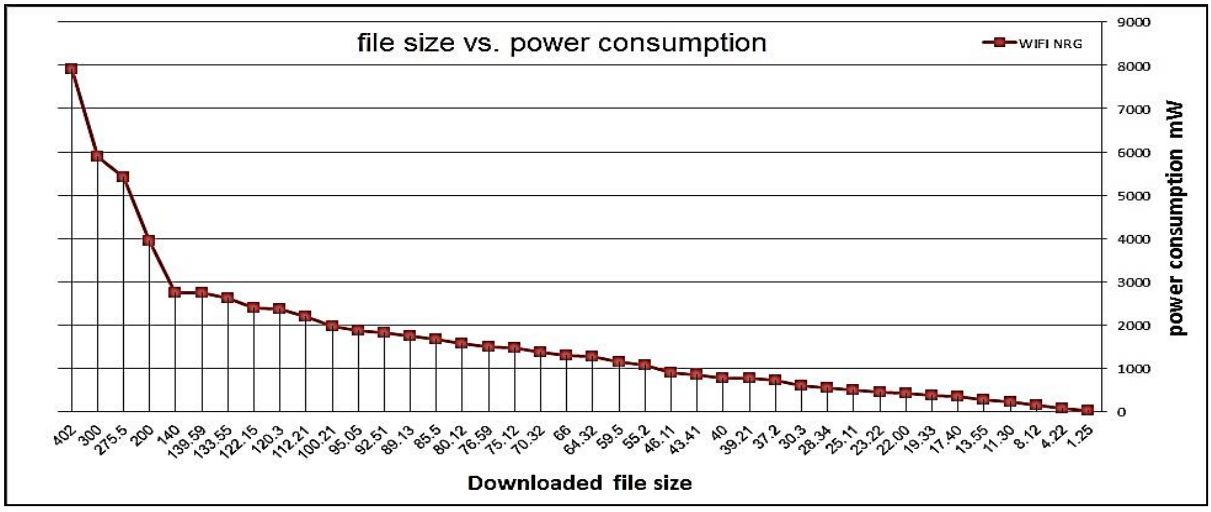

Figure 5: Experimental result using WI-FI connection

4.2.3 Third experiment: Trade-off between No. of training and test sets.

The third experiment was performed using the same files had been used in the previous experiments to be downloaded for testing. Downloading the jobs take the exact first experiment steps but when putting the URLs but using the bottom (download without AHpower) when smart device connected to the internet through 3G cellular network. The results showed anew average of power consumption for five minutes when 2203mW profiled

with nearly the same downloading time (DLT) in the second experiment. Figure 6 shows the third experiment results without using AHpower app with 3G connection clarified that the average of total energy consumption in the smart device for five minutes needed to calculate by new equation. As shown in equation (2), which is very critical to get accurate final results shown Table (4).

$$\text{Energy via 3G} = \frac{\text{Energy consumption value for N of minutes}}{\text{N of minutes} * 60\ seconds} \quad (2)$$

Energy via 3G = 2203/300=7.344mW/sec.

Table 4: sample of downloaded files via 3G

Downloaded Job (file)	Via 3G		
Size in (MB)	Time H,M,S,MS	Speed average Mb/s	Energy mW
1.25	00.00.05.12	2	37.60
4.22	00.00.17.29	2	126.83
8.12	00.00.33.29	2	244.48
11.30	00.00.46.30	2	340.02
13.55	00.00.55.50	2	407.59
17.40	00.01.11.31	2	523.70
19.33	00.01.19.20	2	581.64
22.00	00.01.30.11	2	661.77
23.22	00.01.35.13	2	698.64
25.11	00.01.42.89	2	755.62

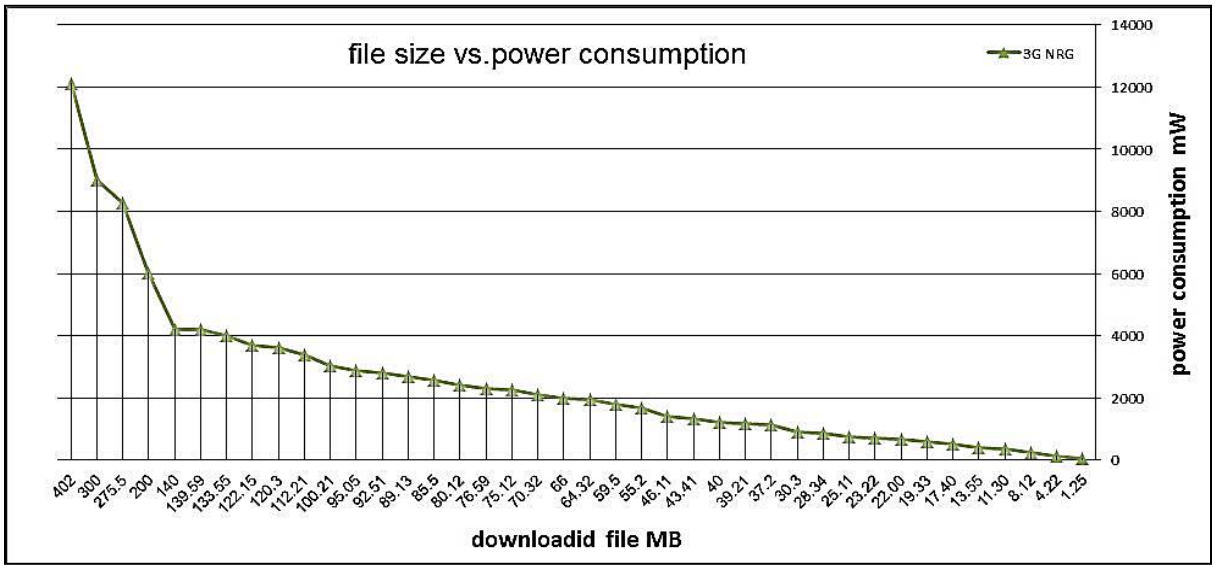

Figure 6: Experimental result using 3G connection

4.2.4 Comparison previous experiments

In AHpower framework … the key of success is the saving in downloading time. When that occurs a power consumption must be decreasing. The first and second experiments are using the same data network to connect internet (WI-FI), but a simple comparison operation declares the different time of downloading jobs results for the same files when the power consumption of WI-FI connection is almost stable with 4.8 mW/sec. Certainly the same could be said about the third experiment even with the used of different data network (3G) which consume 7.344mW/sec when downloading. As explained previously in the details of each experiment. Figure 7 show the linear comparison chart.

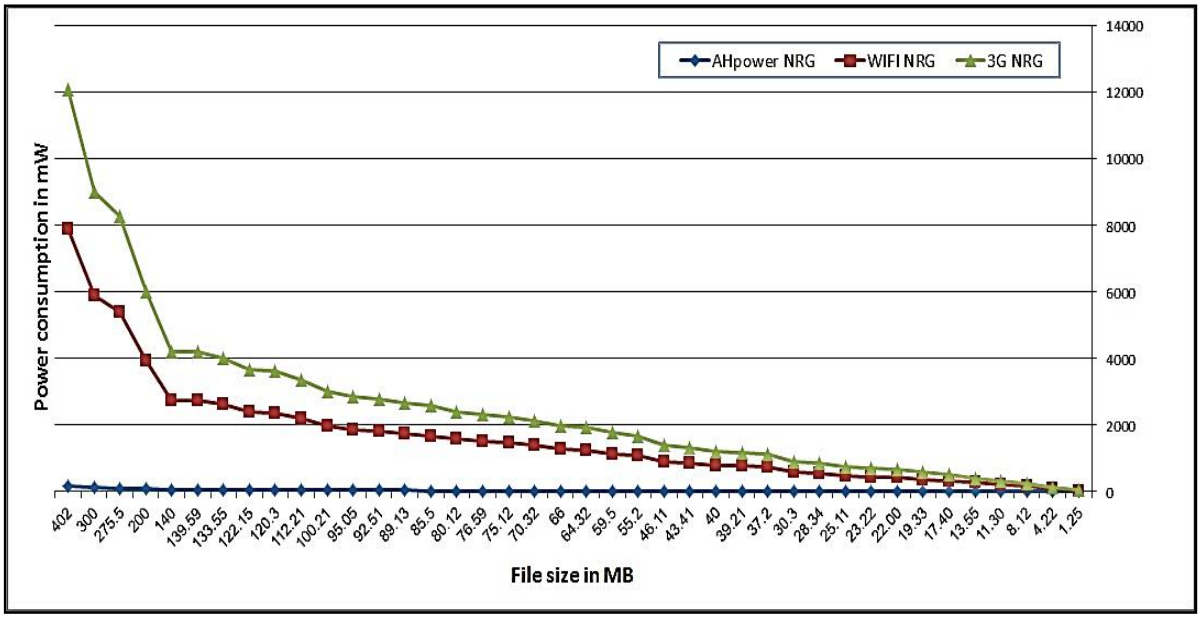

Figure 7: Final experimental comparison chart

Table 5: final power percentage results

Percentage of energy	Mathematical formula	consumption rate	Saving rate
AHpower/WI-FI	(X AHpower - X WI-FI)/ABS (X WI-FI)	2.01%	97.99%
AHpower/3G	(X AHpower - X 3G)/ABS (X 3G)	1.33%	98.67%
WI-FI /3G	(X WI-FI - X 3G)/ABS (X 3G)	65.1%	35.9%

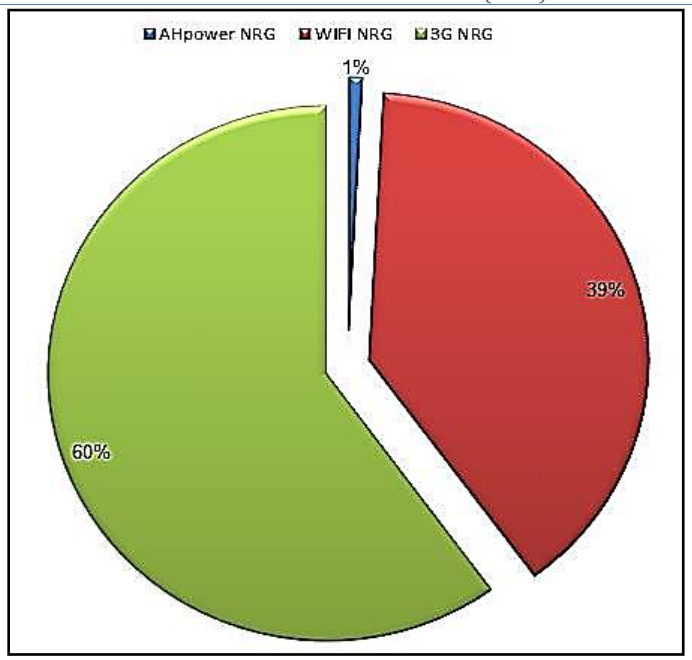

Figure 8: Experiments comparison for power consumption ratio.

5. CONCLUSION AND FUTURE WORK

The final result became clear through all the experiments that are described in detail in the previous section. It confirms the success of the proposed system. Where the energy consumption percentage calculated through the process of downloading the same files (download tasks). When using the proposed system AHpower vs. the same job using WI-FI it reached 2.01% of the original amount of energy, this means that the energy saved is equal to 97.99%. However when the connection occur through the third generation data network (3G), proportion of consumption reached only 1.33% and thus increased energy that was saved to the percentage of 98.67 % as shown below in Table 5. As shown these results characterized the success and the possibility of applying that idea. Many big manufacturers companies that produce smart devices with android operating system can benefit from it as mentioned before in Prospective audience section-chapter one. Figure 8 shows the final consumption percentage ratio for each experiment.

REFERENCES

[1]. G.P. Perrucci, F.H.P Fitzek, J. Widmer, ", Survey on Energy Consumption Entities on the Smartphone Platform", IEEE Computer Society 2011.

[2]. Ahmed E. Youssef ," Exploring Cloud Computing Services and Applications", Journal of Emerging Trends in Computing and Information Sciences,VOL. 3, NO. 6, July 2012.

[3]. Eric Y. Chen, and Mistutaka Itoh," Virtual Smartphone over IP", IEEE Computer Society 2010.

[4]. Aki Saarinen, Matti Siekkinen, Yu Xiao, Jukka K. Nurminen, Matti Kemppainen, and Pan Hui," Can Offloading Save Energy for Popular Apps?", MobiArch'12, , Istanbul, Turkey, August 22, 2012.

[5]. R. Kemp, N. Palmer, T. Kielmann, and H. Balm, "Cuckoo: a computation offloading framework for Smartphones", In Proceedings of MobiCASE, Oct. 2010.

[6]. Dejan Kovachev and Ralf Klamma, "Framework for Computation Offloading in Mobile Cloud Computing", International Journal of Artificial Intelligence and Interactive Multimedia, Vol. 1, № 7, 17.1.2012.

[7]. Eduardo Cuervoy, Aruna Balasubramanianz, Dae-ki Cho, Alec Wolmanx, Stefan Saroiux, Ranveer Chandrax, and Paramvir Bahlx, " MAUI: Making Smartphones Last Longer with Code Offload", MobiSys'10, San Francisco, California, USA, June 15–18, 2010.

[8]. Ying Zhang, Gang Huang, Xuanzhe Liu, Wei Zhang, Hong Mei, Shunxiang Yang," Refactoring Android Java Code for On-Demand Computation Offloading", OOPSLA'12, Tucson, Arizona, USA, October 19–26, 2012.

[9]. Narendran Thiagarajany, Gaurav Aggarwal, Angela Nicoara" Who Killed My Battery: Analyzing Mobile Browser Energy Consumption", WWW 2012, Lyon, France, April 16–20, 2012.

[10]. L.F. Pau" Energy Consumption Effects of WI-FI Off-Loading Access in 3G or LTE Public Wireless Networks", International Journal of Business Data Communications and Networking, 9(2), 1-10, April-June 2013.

[11]. Niranjan Balasubramanian Aruna Balasubramanian Arun Venkataramani," Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications", IMC'09, November 4–6, 2009, Chicago, Illinois, USA.

[12]. Goran Kalic, Iva Bojic and Mario Kusek," Energy Consumption in Android Phones when using Wireless Communication Technologies", MIPRO, 2012 Proceedings of the 35th International Convention, 21-25 May 2012.

[13]. Rahul Murmuria, Jeffrey Medsger, Angelos Stavrou, Jeffrey M. Voas," Mobile Application and Device Power Usage Measurements", Software Security and Reliability (SERE), 2012 IEEE Sixth International Conference - USA, 20-22 June 2012.

[14]. Ahmed Abdelmotalib, Zhibo Wu," Power Consumption in Smartphones (Hardware Behaviorism) ", IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 3, No 3, May 2012.

[15]. Sokol Kosta, Andrius Aucinas , Pan Hui, Richard Mortier, and Xinwen Zhang," ThinkAir: Dynamic resource allocation and parallel execution in cloud for mobile code offloading", 1Google's 2011 Q2 earnings call.

