International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)
ISSN: 0976-1353 Volume 29 Issue 4 – MAY 2022.

Android Battery Saving System

.

J. Jeyaranjani
Department of Computer Science and Engineering,
Kalasalingam Academy of Research and Education,
Anand Nagar, krishnankovil, Tamilnadu, India.

V. Priyadharshini
Department of Computer Science and Engineering,
Kalasalingam Academy of Research and Education,
Anand Nagar, krishnankovil, Tamilnadu, India.

M.Shailesh
Department of Computer Science and Engineering,
Kalasalingam Academy of Research and Education,
Anand Nagar, krishnankovil, Tamilnadu, India.

S. Shreekanth
Department of Computer Science and Engineering,
Kalasalingam Academy of Research and Education,
Anand Nagar, krishnankovil, Tamilnadu, India.

Abstract-This System is an innovative Application Allowing the System and the user to take the usage From Build-in classes and put a list in front of the User for him the full detailed review. The List also Consists of the applications using the battery usage And also determines, analyze the battery level. If the Battery level is low or gets drop down or the consumption of apps is more the system will trigger an alarm telling the user to force stop or close the apps with the permission of the user. This system uses Android Studio as its front end and doesn't use any backend as this type of application doesn't need one since it uses the data from the phone itself and projects to the user. So basically the system helps the user to refrain certain apps to consume more battery power and drain it quickly and user can take some actions on it.

Keywords-Battery Historian, Drop Down level, Android Studio, Permission of user.

INTRODUCTION

Android Battery Saving System is a mobile application which tracks and gather the full top to bottom details of the smart phone battery to provide a better performance for the user. It optimize the battery as much as possible and uses every single bit of the battery which will be very useful for the user who uses the smartphones. Battery saving system provides a simple user interface which also provide a pictorial graphical view of the battery details to the user for understanding the performance and full details of the smartphone battery.

It also intimates the user by trigger a message to user if the battery of the particular smartphone drains as fast as of prediction. This help the user to have a look on it when the problem occurs. It allows the user to user to use only a particular app when he has no time to charge his/her smartphones. It will be the user's battery guardian which looks care of battery as of like the user who take care of his/her smartphones.

By using this user don't need to worry about his battery life and all as he has a safety guard for his/her battery. This allows the user to concentrate on other works instead of having look on the battery life of the smartphones.

Battery consumption for the particular application is more then it will alarm the user regarding this so that the user can close the particular application. People can rely on this application with great ease in order to save the battery of the android phones. The battery level will be determined based on the various applications that are run on the android phone

.

RELATED WORKS

Woo et al. [8] examine caching in their study, as minimal work has been done to optimize the content caching in cellular networks. The increasing number of high speed base stations has made network accessibility more convenient for users. However**,** the problems surfacing with cellular networks is that all user traffic has to pass a limited number of gateways at core networks before reaching the wired internet. Simply increasing the physical backhaul bandwidth is not feasible for

Strategies must be considered. Their study focuses on three types of caching: conventional web caching, prefix-based web caching and TCP-level redundancy elimination. Conventional web caching places information at the digital unit aggregation (DUA) component of the cellular network architecture. However, this approach suffers from two problems. The first problem is that it cannot suppress duplicate objects that are uncacheable or that have different URLs (i.e., aliases)". Secondly , it is difficult to handle handovers from the mobile device to the DNA while the connect is being delivered . prefix-based web caching can overcome the first problem that web caching has, suppressing the duplicate and aliased objects. The drawback of this approach is that its by Woo at al. later in the study. Lastly, TCP redundancy elimination can also handle the issues of traditional web caching, but suffers from a complex implementation and high computational overhead. The first part of the study was to analyze the TCP and application -level characteristics of the traffic, while the second part was comparing the effectiveness of the three types of web caching. Based on the results, 59.4% of the traffic is redundant with TCP-level redundancy elimination if we have infinite cache. In regards to caching options, standard achieved 21.0.27.1% bandwidth savings with infinite cache, while prefix-based web caching produced 22.4-34.0% bandwidth savings of 26.9-42.0% with only 512 GB of memory cache.

Li et al. [9] perform an analysis of energy consumption on android smartphones, focusing on how the device is used as opposed to the application running. To maintain consistency, an additional battery with a fixed voltage source is attached to a smartphone instead of using the traditional lithium-ion battery. A series of test cases are them performed on three different android devices, and the electric current is measured with a multimeter. In each device's test case, the smartphone was 50% brightness, and all applications were closed.

Hoque at al. [10] Present an analysis of the battery in order to examine charging mechanisms, state of charge estimation techniques, battery properties, and the charging behaviour of both devices and users using data collected from the Carat [11] application.. carat is an application that tracks the application you are using but does not measure energy consumption directly. The first analysis examines the charging techniques of smartphones and the charging rate. The charging mechanism that are variants of the established CC-CV and DLC methods are identified The second part of the analysis examines battery properties such as the changes in its capacity, temperature when charging, and the battery health. The results indicated a linear relationship between the remaining battery capacity and final voltage, and a decrease in batter temperature over time as the device charged. In addition, the health of the battery did nit indicate increases in battery temperature.

Kim et al. [12] Discuss the differences between battery and energy consumption, explaining how they are not always equal. When the battery discharges, portion of the stored energy become unavailable. Energy-saving techniques do not take this measurement into account, leading to incorrect calculations, Kim et al. propose that battery consumption should be the metric considered when proposing a savings plan. They design an application to calculate battery consumption, and evaluate their model with a series of test cases. The test cases include many power hungry application, but their consumptions rates and periods of activity differ. The analysis examine the relationship between the systemwide power consumption and unavailable energy. In the initial trial, an increase in power consumption also increased the unavailable energy, and in some cases reduced the actual delivered energy by over 50%. When applying the measurements technique to the test cases with scaling governors that mange CPU frequency and voltage , certain scenarios even indicated battery consumption can decrease when energy consumption increases.

Lee et al. [13] focus specifically on battery aging, and the importance of quantifying the process. They propose an online scheme that tracks battery degradation without the use of any external equipment. The scheme functions by logging the amount oof time required to charge the battery. A set of different lithiumion batteries with different ages are measured to set a baseline charges time. The focus on the analysis is based on the middle region, charge levels approximately between 40%-80% this is due to the linear charge rate in the given period. To calulate the battery the the0retical charge time of the region, and uses the actual charging time of the given range. The accuracy of the efficiency measurements were 0.94+/-0.82 to 0.99.

Bulut et al. [16] have also utilized prediction in a unique way, creating a crowdsourced line wait-time monitoring system with smartphones. Its implementation at grocery stores,
DMV's and banks would allows users to make informed choices in time-sensitive environments.
Know as lineking, it has been tested at a coffee shop at the state university of new York at buffalo. Customers who enters the shop will establish a connection with the service, where any connection lasting longer than 2 minutes but less than 20 is deemed a customer. The wait-time calculation is completed on the server side of the application, taking the time of the day, the day of the week, and seasonality into account. The estimated wait times are accurate within 2-3 minutes.
Lineking is comprised of two components, a client-side and a server. The client side is comprised of three subcomponents: phone-state-receiver, wait-time-detection, and datauploader. The phone-state-receiver is comprised of a variety of receivers registered to monitor various events for the application. The most notable event is the device entering and exiting the shope. The wait-time-detection component can use either location sensing or WIFI sensing to detect the user's presence at the shop. In order to preserve battery life, the component begins monitoring the device under two conditions: if the user opens the application to check the waittime or if the user is physically close to the shop. Once a condition is triggered, the application begins to monitor the user's location. For location sensing, if the user is within a specific range of the shop, the application will set a proximity alert to register the timestamp of entering the shop. If they are outside of the specified range, the application will estimate the arrival time of the user, and recheck their location at that time. If the user does not travel towards the shop after a certain amount of time, the monitoring will cease. In the Wi-Fi sensing approach, the application will monitor Wi-Fi beacons periodically to determine when the user enters and leaves the shop. Their calculation in this approach takes into account the time delay of the scanning period. Lastly, the data-uploader component is responsible for uploading the wait-times to the estimation system.

Rattangan et al. [19] examine prediction and battery life together, evaluating online power estimations from battery monitoring units. They discuss the current methods of online and offline monitors, indicating the pros and cons of each. While online methods are more feasible and scalable, their results have a higher error rate due to three problems that are not taken into consideration: battery capacity degradation, asynchronous power consumption behaviour, and the effect of state of charge difference in hardware training. The battery capacity of a device will decrease after usage, while online methods use the original battery capacity value without taking this into account. Asynchronous power consumption refers to readings where power consumption is misattributed to a given component or resource. Lastly, the effect of state of charge (SOC) difference in hardware training refers to the power consumption estimation of the hardware at different battery percentages. While the consumption rate should be uniform regardless of the state of charge, that is not the case for online battery monitoring units.

Anguita et al. [21] attempt to use machine learning to overcome battery limitations. They propose sensors can be used to predict the actions of the user. They use an existing machine learning framework and modify it to meet the

resource constraints of a smartphone. Their implementation is then validated in a series of test cases where the framework predicts whether the user is walking, walking upstairs, walking downstairs, sitting, standing, or laying. While the test cases are outside the scope of traditional application monitoring, they illustrate the usage of machine learning is feasible for mobile devices, and the resource requirements can be reduced.

METHODOLOGY:

Date	Percentage	App01	App02	App03	App04
04/09/2017 10:24	3	5	4	5	
04/09/2017 10:29	2	5	5	0	
04/09/2017 10:34	3	5	5	0	
04/09/2017 10:39	2	5	5	5	
04/09/2017 10:49	2	5	5	5	
04/09/2017 10:54	2	5	5	5	

Test Case
50% brightness screen
Opening GPS
Opening Wi-Fi
Wi-Fi Download (2.55 Mpbs)
GSM Download (35kpbs)
Opening Bluetooth
Bluetooth Searching Devices
Bluetooth sending data
CPU Single thread
CPU multithreads
CPU stress condition
Opening terminal
Calling
Incoming call
Sending a message
Taking a picture
Playing music
Playing video

Table 1: Sample dataset for IPF example

Date	Percentage	App01	App02	App03	App04
04/09/2017 10:24	3	5	4	5	

Table 2: First row of Data

app01	app02	app03	app04	app05	app06
1 1	1	1	1	1	1

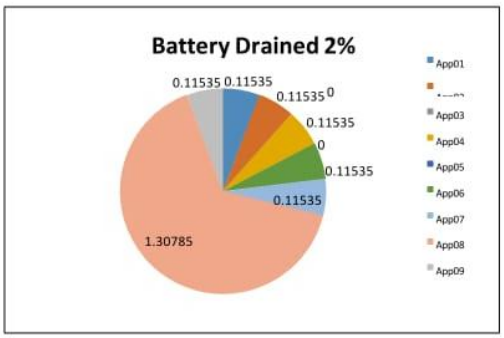
Table 3: Initial weight/consumption rate of applications

$t = \sum (a_i * m_i)$

$t = 0.0882 * 5 + 0.0882 * 5 + 0.0882 * 5 + 0.0882 * 5 + 0.0882 * 5 + 1 * 5 + 0.0882 * 5$ $t = 7.646$

$a_y^1 = p_x * (a_y / t)$

$app_{01} = 2 * (0.0882 / 7.646) = 0.02307$ $app_{02} = 2 * (0.0882 / 7.646) = 0.02307$ app_{03} = not running
$app_{04} = 2 * (0.0882 / 7.646) = 0.02307$ app_{05} = not running
$app_{06} = 2 * (0.0882 / 7.646) = 0.02307$ $app_{07} = 2 * (0.0882 / 7.646) = 0.02307$ $app_{08} = 2 * (1 / 7.646) = 0.26157$
$app_{09} = 2 * (0.0882 / 7.646) = 0.02307$



The estimated total consumption of the active applications is calculated by multiplying each active application's estimated consumption rate by the number of minutes it is active in this reading. This information is required in order to determine the updated consumption rates, which are calculated on a per-minute ratio.

Let t represent the sum of the estimated consumption rate for all active applications

Let a represent the estimated application consumptions in a given reading

Let m represent the number of minutes each application was running in a given reading $t = \sum (ai * mi)$ $t = 1 * 5 + 1 * 4 + 1 * 5 + 1 * 5 + 1 * 5 + 1 * 5 + 1 * 5$ $t = 34$

The battery percentage consumed by all open applications in the timestamp is 3%. The following formula is used to determine the percentage that each individual application consumed.

Let px represent the total percentage of battery consumed in a given reading Let ay represent the estimated application consumption

$ay1 = px * (ay / t)$

$app01 = 3 * (1 / 34) = 0.0882$ $app02 = 3 * (1 / 34) = 0.0882$ $app03 = 3 * (1 / 34) = 0.0882$ $app04 = 3 * (1 / 34) = 0.0882$ $app05 = 3 * (1 / 34)$ = not running $app06 = 3 * (1 / 34) = 0.0882$ $app07 = 3 * (1 / 34) = 0.0882$ app08 = not running $app09 = 3 * (1 / 34) = 0.0882$

Figure 14: Visual representation of how much each application contributed to the 3% drain

app01	app02	app03	app04	app05	app06
0.0882	0.0882		0.0882		0.0882
1	0.0882				

Table 5: Weight/consumption rate of applications after one iteration of IPF

Date	Percentage	App01	App02	App03	App04
04/09/2017 10:29	2	5	5	0	

Table 6: Second Row of Data

With the second row of data, the battery percentage consumed by all open applications is 2%. The same formula is used to determine both the estimated total consumption and percentage that each application used.

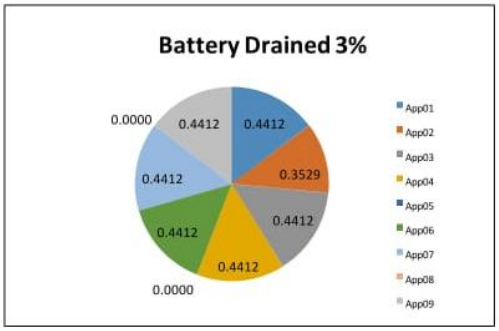
$t = \sum (ai * mi)$

$t = 0.0882 * 5 + 0.0882 * 5 + 0.0882 * 5 + 0.0882 * 5 + 0.0882 * 5 + 1 * 5 + 0.0882 * 5$ $t = 7.646$

$ay1 = px * (ay / t)$

$app01 = 2 * (0.0882 / 7.646) = 0.02307$ $app02 = 2 * (0.0882 / 7.646) = 0.02307$ app03 = not running $app04 = 2 * (0.0882 / 7.646) = 0.02307$ app05 = not running $app06 = 2 * (0.0882 / 7.646) = 0.02307$ $app07 = 2 * (0.0882 / 7.646) = 0.02307$ $app08 = 2 * (1 / 7.646) = 0.26157$

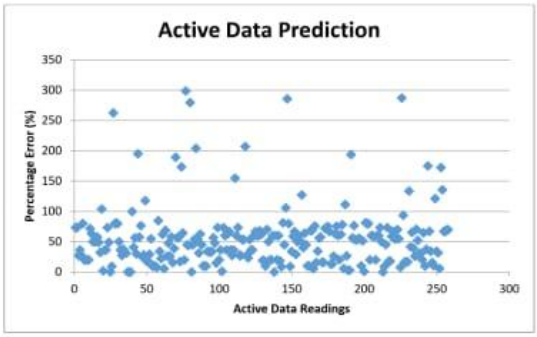
app09 =

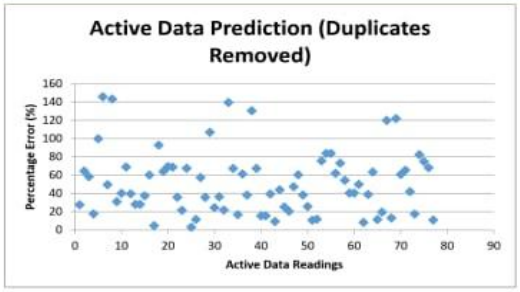


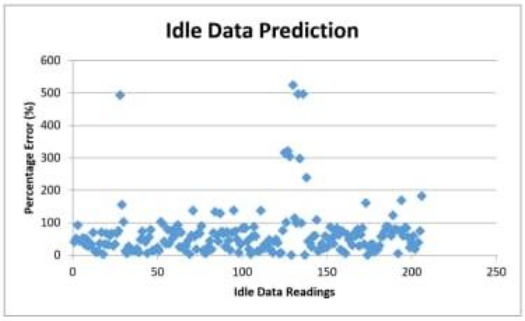
2 * (0.0882 / 7.646) = 0. 02307

Figure 7: Visual representation of how much each application contributed to the 2% drain, based on the updated weight/consumption values

app01	app02	app03	app04
0.00252	0.00252	0.0882	0.00252







CONCLUSION:

As applications and smartphone devices become increasingly powerful, battery life remains a large problem for users. Smartphones are capable of integrating many aspects of a user's life, leading us to become more dependent on them. As such, it is crucial they remain powered throughout a user's entire day, leading to research and examination on this topic. The current implementations provide the changes that need to be made, but rely on repeated human interaction. As people may forget and not be vigilant in these changes, they are not used efficiently. The proposed application would be a first step in overcoming these challenges and automating this process.

REFERENCE:

[1] B. Reed, "A Brief History of Smartphones," PC World, 18 June 2010. [Online]. Available: http://www.pcworld.com/article/199243/a_brief_history_of_smartphones.html#slide1. [Accessed 17 June 2017]. [2] CAT, "New Research Reveals Mobile Users Want Phones To Have A Longer Than Average Battery Life," CAT, 28 November 2013. [Online]. Available: http://catphones.com/news/pressreleases/new-research-reveals-mobile-users-want-phones-to-have-a-longer-than-average-batterylife.aspx. [Accessed 19 May 2015]. [3] A. Pathak, A. Jindal, Y. C. Yu and S. P. Midkiff, "What is Keeping my Phone Awake? Characterizing and Detecting No-Sleep Energy Bugs in Smartphone Apps," in 10th International Conference on Mobile Systems,

Applications, and Services, Low Wood Bay, 2012. [4] F. Xu, L. Yunxin, T. Moscibroda, R. Chandra, L. Jin, Y. Zhang and Q. Li, "Optimizing Background Email Sync on Smartphones," in 11th Annual International Conference on Mobile Systems, Applications, and Services, Taipei, 2013. [5] F. Richter, "Statista," 2 April 2014. [Online]. Available: http://www.statista.com/chart/2082/topsmartphone-apps-2013/. [Accessed 15 June 2015]. [6] A. Albasir, K. Naik, B. Plourde and N. Goel, "Experimental Study of Energy and Bandwidth Costs of Web Advertisement on Smartphones," in 6th International Conference on Mobile Computing, Applications and Services, Austin, 2014. [7] F. Qian, S. Sen and O. Spatscheck, "Characterizing Resource Usage for Mobile Web Browsing," in 12th Annual International Conference on Mobile Systems, Applications, and Services, Bretton Woods, 2014. [8] S. Woo, E. Jeong, S. Park, J. Lee, S. Ihm and K. Park, "Comparison of Caching Strategies in Modern Cellular Backhaul Networks," in 11th Annual International Conference on Mobile Systems, Applications, and Services, Taipei, 2013. [9] X. Li, X. Zhang, K. Chen and S. Feng, "Measurement and Analysis of Energy Consumption," in 4th IEEE International Conference on Information Science and Technology, Shenzhen, 2014. [10] M. A. Hoque and S. Tarkoma, "Characterizing Smartphone Power Management in the Wild," in ACM International Joint Conference on Pervasive and Ubiquitous Computing, New York, 2016. 70 [11] "Carat," Carat Team, 12 March 2018. [Online]. Available: https://play.google.com/store/apps/details?id=edu.berkeley.cs.amplab.carat.android&hl=en. [Accessed 22 May 2018]. [12] M. Kim, Y. G. Kim and S. W. Chung, "Measuring Variance between Smartphone Energy Consumption and Battery Life," Computer, pp. 59-65, 16 August 2013. [13] J. Lee, Y. Chon and H. Cha, "Evaluating Battery Aging on Mobile Devices," in 52nd Annual Design Automation Conference, New York, 2015. [14] B. D. Higgins, K. Lee, J. Flinn, T. Giuli, B. Noble and C. Peplin, "The Future is Cloudy: Reflecting Prediction Error in Mobile Applications," in 2014 6th International Conference on Mobile Computing, Applications and Services (MobiCASE), Austin, 2014. [15] G. Kotsev, L. T. Nguyen, M. Zeng and J. Zhang, "User Exercise Pattern Prediction through Mobile Sensing," in 6th International Conference on Mobile Computing, Applications and Services (MobiCASE), Austin, 2014. [16] M. F. Bulut, Y. S. Yilmaz, M. Demirbas, N. Ferhatosmanoglu and H. Ferhatosmanoglu, "LineKing: Crowdsourced Line Wait-Time Estimation Using Smartphones," in 2012 4th International Conference on Mobile Computing, Applications and Services (MobiCASE), Seattle, 2012. [17] D. Gordon, S. Frauen and M. Beigl, "Reconciling Cloud and Mobile Computing using Predictive Caching," in 2013 5th International Conference on Mobile Computing, Applications and Services (MobiCASE), Paris, 2013. [18] Y. Li, B. Luo, H. Chen and W. Shi, "One Charge for One Week: Hype or Reality?," in International Green Computing Conference (IGCC), Dallas, 2014 . [19] E. Rattagan, E. T. Chu, Y.-D. Lin and Y.-C. Lai, "Semi-Online Power Estimation for Smartphone Hardware Components," in 10th IEEE International Symposium on Industrial Embedded Systems (SIES), Siegen, 2015. [20] E. Peltonen, E. Lagerspetz, P. Nurmi and S. Tarkoma, "Energy Modeling of System Settings:," in IEEE International Conference on Pervasive Computing and Communications (PerCom), St. Louis, 2015. [21] D. Anguita, A. Ghio, L. Oneto and S. Ridella, "Smartphone Battery Saving by Bit–Based Hypothesis Spaces and," in International Joint Conference on Neural Networks (IJCNN), Beijing, 2014. [22] "DU Battery Saver - Battery Charger & Battery Life," DU APPS STUDIO, 22 May 2018. [Online]. Available: https://play.google.com/store/apps/details?id=com.dianxinos.dxbs&hl=en. [Accessed 22 May 2018]. 71 [23] "Battery Doctor(Battery Saver)," Cheetah Mobile Inc., 18 May 2018. [Online]. Available: https://play.google.com/store/apps/details?id=com.jinshan.kbatterydoctor_en. [Accessed 22 May 2018]. [24] S. V. Rajaraman, M. Siekkinen and M. A. Hoque, "Energy Consumption Anatomy of Live

Video," in CANDARW: The Fifth International Symposium on Computing and Networking Workshops, Aomori, 2017. [25] M. Brocanelli and X. Wang, "Making Smartphone Smart on Demand for Longer Battery Life," in IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Atlanta, 2017.