Android Games Analysis – Why does my playtime reduce my talk time?

Milind Gokhale
Indiana University
+1-812-369-7654
mgokhale@indiana.edu

Renuka Deshmukh
Indiana University
+1-812-606-2884
renudesh@indiana.edu

ABSTRACT
Mobile games are heavy and consume huge amount of memory (RAM), Storage Space, Data and most importantly Battery. This makes them kill the phone battery and reduce the talk time of the phone which results in smartphones not lasting even one full day of charge. The light casual and social games are connected to internet all the time and also cause battery drain on phones. In this project we intend to study the smartphone resource consumption by games and come up with the main issues which cause battery drain. Our study identified many issues in Games and showed that some of the popular games today fail to keep in mind some simple things like avoiding wakelocks and closing connections prompty owing to which they end up sucking more power from the phone and more data usage for the user.

Categories and Subject Descriptors
D.2.5 [**Software Engineering**]: Testing and Debugging; D.2.8 [**Metrics**]: Performance Measures; D.2.10 [**Design**]: Methodologies; C.4 [**Performance of Systems**]: Measurement Techniques;

General Terms
Mobile Computing, Measurement, Documentation, Performance, Design, Economics.

Keywords
Smartphone Applications, Radio Resource Utilization, Keywords are your own designated keywords.

1. INTRODUCTION
Now-a-days smartphones have become very common in households. Almost every individual uses a smartphone now and as per a study by COMSCORE [1], Games account for 16% of the total time spent on mobile apps in the US. Mobile games are one of the biggest source of mass entertainment these days. It appeals to a diverse group of audience ranging from young to old people across the world. Growth is because of the emergence of modern technology and higher configuration in smartphones as a platform for games and innovative games that attract a lot of people. As per a study by Juniper Research [3], Mobile game revenues will reach $28.9Bn by 2016, up 38% from 2014. 64% of users who downloaded apps have downloaded a game (as shown in figure 1). Among most frequent gamers, social games are now the most popular genre, and are increasing in popularity continuously. Social games and puzzle, board games account for 77% of the played games on smartphones.

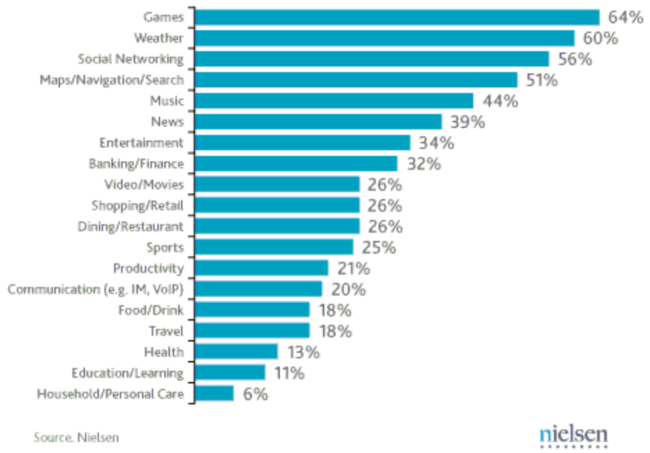


Figure 1: Most Popular Mobile App Categories [6]

However, mobile games consume huge amount of memory, Storage Space, Data and most importantly Battery. This reduces the talk time of the phone drastically. According to utility developer Cheetah Mobile, today's top mobile games are often one of the biggest drains on the performance of iOS and android devices and games like Bejeweled and Candy Crush Saga require lot of storage space and huge amount of memory (RAM) when running. This results in the poor performance of the battery than if the phone is used for just texting and calls.

This area is relatively unexplored with very few prior studies that evaluate the adverse effects of high gaming on smartphones. This may be because there are innumerable genres of games and numerous games in every genre. So the categorization of games becomes difficult based on liking, age group and location. Moreover every genre of games has its own possible resource requirements and limitations. Our study points out some major issues in today's games due to which the games end up using more resources on smartphones.

In section 2 we mention some of the work related to our study done till now, in section 3 we explain our methodology of testing and evaluation in detail. In section 4 we give our evaluation and findings from the tests done and in section 5 we discuss some of the possible future enhancements and implementation.

2. RELATED WORK

Recently some amount of attention is being given on the issue of battery drain due to games by certain service providers and organizations. [2] Cheetah Mobile – a mobile utility developer also did a study on mobile games in which they tested out games like Bejeweled, Candy Crush Saga, FarmVille etc. and checked to see how much memory they use, how much storage space is required and how much unnecessary data is kept by each game on the smartphone. As a solution, Cheetah Mobile suggested users to clean up junk files of their phone and to close their games when they're not using them.

Service provider Verizon Wireless also keeps a watch on the detrimental apps for smartphones and maintains a list of high risk apps on their website [4] which contains apps which can have negative effects on user's devices like loss of functionality, unexpected high data usage, battery draining 2 or 3 times faster than normal, and security and privacy exposure. Verizon also suggests android users to close these high risk apps when not in use.

These studies although focus on effects of games on devices, they fail to explore the root cause of inefficiencies and bad practices or negligence in games development. Our study goes beyond and tries to find the root cause of inefficiencies in games and suggest recommendations to game developers. In [5] the study conducted by Zhang et al. characterized the energy and bandwidth efficiency of instant messaging apps which require constant user interaction like gaming apps. In a similar approach we study the characterization of two popular gaming apps in this paper.

3. METHODOLOGY

In this section we describe the approach for data collection and analysis. For this we d

3.1 Game States

We evaluate two states of game play – 1. When the game is running in foreground and 2. When the user has switched to another app and the game is running in background.

3.1.1 Foreground

This is the state when the gamer is playing the game or the game is on the foreground of the apps running on the smartphone.

3.1.1.1 Idle

Even though smartphones are with the people all the time, people do a lot of things off the smartphone as well. Hence there are numerous possible occasions when the user has started the game however has switched on to some other work. This typically can happen in transit during travelling, in commute or when the user falls asleep without manually closing the game.

3.1.1.2 Playing

This is the active state when the user is actually playing the game. This is expectedly the most resource consuming state. We measure the resource consumption of the game in this state in 3 scenarios –

- No Internet: When the user is playing the game, however the phone is not connected to the internet.
- Wi-Fi: When the user is playing the game and the phone is connected to a Wi-Fi network.
- LTE: When the user is playing the game and the phone is on the LTE network.

3.1.2 Background

This is the state when the user started the game, however switched to the other application without closing the game. This state can also occur when the game is ON but the phone is in sleep mode. We measure this state with the expectation that the game must consume minimal resources when running in background.

3.2 Experiment Setup

We use a rooted Samsung Galaxy S2 LTE smartphone running on Android Jellybean 4.3.2 OS for collecting data specific to games running with no internet and Wi-Fi. For LTE based tests, we use HTC One M7 running on Android Lollipop 5.0.2 OS and connected to AT&T service provider. For data collection and offline data analysis we used HP Envy Laptop running on Windows 8.1 operating system. We installed the USB drivers of the respective smartphones, android SDK and AT&T ARO data analyzer on the laptop. AT&T ARO data collector apk was installed on both the phones. To analyze the network traffic dump we installed Wireshark and Microsoft Network Monitor.

3.3 Games

3.3.1 Farm Heroes Saga

Farm Heroes Saga is one of the most popular games in Google Play Store. It is a social game which can be played both online as well as offline. This gave us an opportunity to execute the game in both environments with and without internet connection and hence compare and study the resource consumption in both these setups. It is also a graphics intensive game which could help us monitor its energy consumption due to the CPU usage.

3.3.2 2048

2048 is also a highly popular game on Google Play Store. It also supports both online and offline versions. However unlike Farm Heroes Saga, 2048 is a casual game which does not require heavy graphic rendering. We expect this game to give better resource consumption values.

3.4 Tools

3.4.1 ARO

ARO is a free diagnostic tool which can pinpoint the source of wasteful data and power drains. It also gives useful suggestions about the ways for fixing these issues. Majorly, ARO will be used for capturing game-related traces from the smartphone and analyze the associated TCP connections and data traffic.

3.4.2 Wakelock Detector

Wakelock detector will be used to detect wakelock issues in the games. Wakelock Detector app needs root permissions to execute. This app is useful in detecting no-sleep bugs as suggested in [10].

3.4.3 Power Tutor

Power tutor tool gives 5 minute duration graphs for CPU, Memory, Screen rotation and GPU. These details will help measure and monitor the effects of the games on the resources of the smartphone.

3.4.4 Usemon

Usemon is a simple application for monitoring usage of system resources. It displays graphs of CPU, RAM, and network usage. This app does not require rooted permissions to monitor data related to system resources [9].

3.5 Test Subjects

Initially we planned to collect the data from multiple users playing games on their smartphones and later perform offline analysis of this data on laptop. However, ARO needs rooted phone for remote data collection which may not be available with multiple users. ARO also supports data collection on a phone without rooting via the USB debugging mode. So we are using the USB debugging mode for our project. For USB debugging mode, the phone needs to be connected to the laptop via cable during the test. Moreover the resource consumption of the game is not dependent upon expertise of player playing the game. Hence, test subjects will not be required for specifically collecting data.

3.6 Test Cases

We identified the test cases to be covered in the current scope of our project. Each test was recorded over duration of 5 min. These test cases are listed in the following table.

Table 1: Test Cases

	Description	Network Channel
1	Game Screen On, Idle	No Internet/Wi-Fi/LTE
2	Game Playing	No Internet/Wi-Fi/LTE
3	Ad Statistics	Wi-Fi/LTE
4	App in background	-

3.7 Testing Steps and Analysis Approach

To obtain a comprehensive view of the game resource consumption, we test the game execution in different identified states by using AT&T ARO tool. We collect data for each state in various settings of the phone like when the phone is not connected to internet (NoInternet), when the phone is connected to Wi-Fi and when the phone is connected to LTE network. For every ARO test we connect the phone to the laptop via USB cable in USB debugging mode and execute the test for about 5 mins duration. Then we start the ARO video recording on laptop and begin playing the game on phone. Next we analyze the test report produced by ARO for any issues reported by ARO and look into the data traffic statistics, energy consumption statistics, RRC states and Trace Graph produced by the ARO profiler.

Before beginning the test, we close all the open applications and background applications using Greenify. For NoInternet tests the phone has to be connected to internet (via LTE/Wi-Fi) when starting ARO test. Once the test begins successfully and video recording for the test is visible, we disconnect the phone from internet connection. This is a required step because in order for the ARO analysis engine to start the VPN connection on the phone, it must be connected to internet. In order to isolate the game statistics from other applications, we identify the IP addresses in the *domains accessed* section of the ARO analyzer.

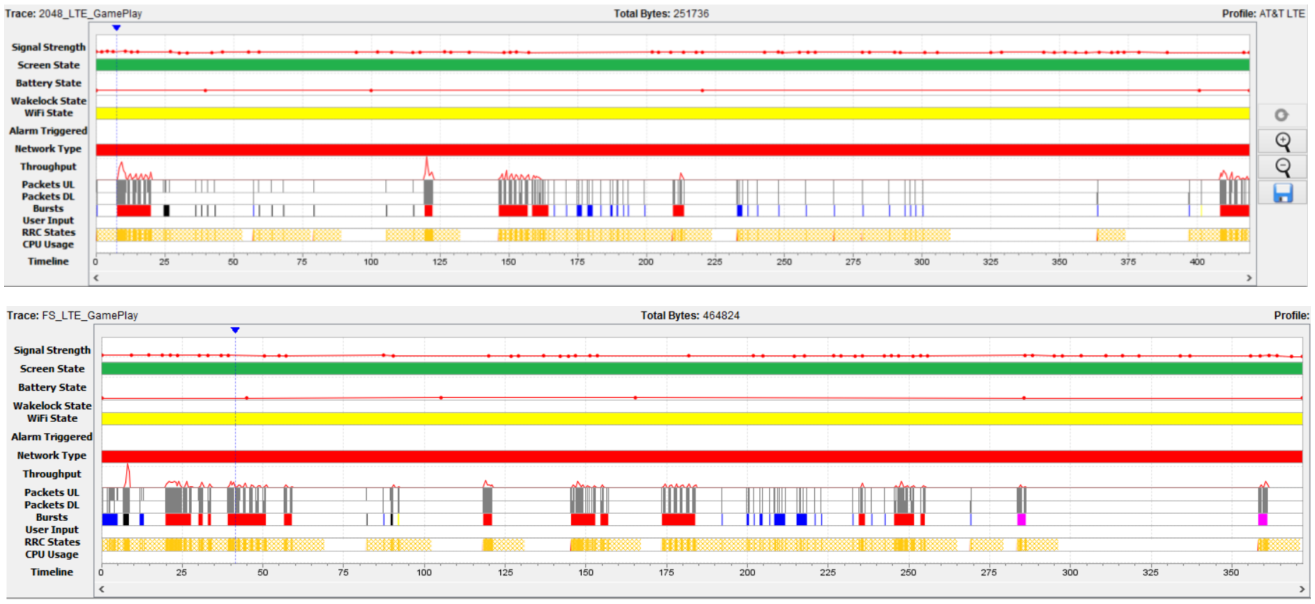


Figure 2: Diagnostics chart for LTE tests on (a) 2048 and (b) Farm Saga

3.8 Metrics

We mainly use three metrics to measure the network traffic, data transfer and the energy consumption and efficiency. (i) We measure energy consumption as shown by ARO tool in Joules (J), (ii) Energy efficiency is measured in Joules/second (J/min) (iii) Data transfer rate is measured in bytes/s. (iv) The test duration is measured in minutes.

4. FINDINGS

4.1 Idle

Keeping the screen ON consumes a lot of battery owing to the LCD display requirements of the phone. For efficient battery consumption, Android OS consists of settings for turning the screen OFF after a certain amount of time has passed without any user interaction on the screen. In all the *idle* state tests for Farm Saga, it was noted that Farm Saga overrides default screen timeout and keeps screen in ON state even when the game is idle. Farm Saga drained 2% battery when run in idle mode. At this rate the entire battery will drain out in under 5 hours. About 210 J of energy is consumed during the *idle* tests for Farm Saga. Total screen energy consumed remains almost constant whether the phone is connected to internet or not. This causes the battery to drain a lot faster and results in the phone not lasting for a long time. We further analyze the root cause of this issue and discuss our findings in section 4.5.

The Radio Resource Control (RRC) in UMTS state machine's main purpose is to efficiently utilize limited radio resources and to improve the phone battery life [18]. Typically there are three RRC states: IDLE, CELL_DCH, and CELL_FACH. The radio power consumption at DCH is the highest (600 to 800 mW). The FACH state is an intermediate state between IDLE and DCH with only low-speed shared channels allocated to a handset. Radio power at FACH is 55% to 75% of that at DCH. LTE on the other hand has only two states- IDLE and Connected. Connected is the high powered states, hence LTE consumed a lot of energy for even small data transfers, and reduce battery life. We found in the idle test for Farm Saga that even though the phone was not connected to internet, the RRC state of the phone remains in the DCH state for 38.91% and in FACH state for 43.66% of the time which consumes more battery unnecessarily.

If an app opens too many simultaneous TCP connections, the size of each connection will be smaller. This means that throughput will be limited, performance will decrease, and energy will be wasted. In Farm Saga as well as 2048 tests, on multiple occasions even when the phone was idle and connected to LTE or Wi-Fi there were multiple sets of bursts found in the ARO diagnostics trace. Whenever possible, it is good practice to group requests together in order to improve performance, save energy and reduce bandwidth. Since a single request for more data is likely to provide a better user experience than several smaller requests, bundling, or batching up multiple requests at the application level is recommended [14]. This can be done with persistent connections. Another technique for improving application performance is HTTP pipelining. If an application is running on a client that supports persistent connections, it can "pipeline" its requests [14].

Recommendation: Whenever possible, multiple connection bursts should be grouped together using persistent TCP connections and HTTP pipelining.

4.2 Playing

4.2.1 *Connection Closing*

If the connection is not closed by design, it will likely be closed by a time-out as part of connection management. However this requires a timeout and thus energy is wasted in managing these connections. In the tests on Farm Saga it was found that on an average more than 25% of the total consumed energy (16.9 J – 84.5 J) is wasted in controlling the connections that were not closed by the server promptly. Even though 2048 is not a social game and requires low graphics rendering, it wasted an average of 50% energy (37.9 J – 161 J) in controlling unclosed connections. Figure 2 shows that the phone remains in active state of RRC consuming more energy because the open connections are not closed promptly.

Recommendation: The TCP connection should be closed quickly after the data is transferred to prevent leaving radio channels open.

4.2.2 *Duplicate TCP Content and Cache Control*

Duplicate requests and TCP content is a typical issue in games and websites. It has several effects on the phone like more bandwidth consumption and battery drain because of multiple requests for downloading the same content [16]. Some applications do not make use of caching mechanisms which also leads to unnecessary downloads that waste energy and make the application slower as well as increases the data usage of the user. During the tests of Farm Saga it was noted that there was 29% duplicate TCP content transferred. Also 50% of the times cache headers were absent which resulted in the files being downloaded in duplicate manner.

Recommendation: Cache Control should be implemented properly in the applications.

4.3 Background

The expected result when the game is put in background is that it does not make any requests to the game or ad server. 2048 made no calls to any ad or game server when the game was put in background. But interestingly, Farm Saga continued to make calls to the game server even though the app was put in background. This resulted in unnecessary data and battery usage, thereby reducing the phone battery life and increasing the data usage. Farm Saga made a total of 17 calls to the game server and 6 calls to the Facebook server in 5 mins duration even though the game was running in background.

Recommendation: When the game has been put to background, all requests to the game and ad server should be stopped, thereby conserving battery and data usage of the users' phone, and further improving the performance.

4.4 Ad Statistics

Most of the games on Google Play Store are free games. In order to gain revenues, game companies collaborate with ad providers because they pay the developers for the ads displayed and the ads clicked by the user [11]. This leads to a lot of inefficiencies in the performance of the game as discussed below.

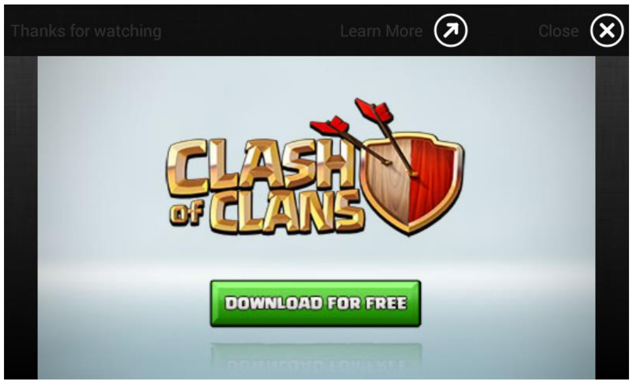


Figure 3: Ad after 2048 gameplay

Sending multiple files over the internet without compression consumes bandwidth and energy **[**12**]**. In multiple tests for 2048, ARO recorded about 10-15 KB of files were sent without compression. In order to make efficient use of bandwidth, the files sent over the network should be compressed. Moreover zipping files on server and unzipping files on the client is a very low cost operation. Many text files contain excess whitespace to allow for better human coding. Files like javascript files, HTML, CSS and JSON consume a lot of time to download when not minified. Running these files through a minifier removes the whitespaces and reduces file size. As per ARO reports, during the *Idle* state test of Farm Saga on LTE network, a total of 24 KB would have been saved by minification of files. All these files were related to Ads.

The CSS rule "display: none;" is used to hide HTML objects from being shown on a page. However, this does not prevent the objects from being downloaded to the mobile device. This increases the overhead for an app and slows down rendering especially in the mobile networks where bandwidth is constrained. In the multiple tests for 2048, it was found that ARO discovered files with CSS styling "display:none". On analysis of these files it was found that they were generated by Ad servers like ad.mopub.com, ad.doubleclick.net and gcdn.2mdn.net. Such content which need not be displayed on the page should be removed from the HTML thus reducing the size of the HTML and decreasing the bandwidth overhead.

It was noted that even when the game is being played by the user (game in foreground), the game calls the Ad servers to fetch ads which were never shown to the user. Ads were only visible once the game had ended (figure 3). As shown in table 2 (highlighted rows), 2048 sent 26 requests to various ad servers requesting for ads in a span of just 5 mins. Since the user was playing the game he never saw these downloaded ads. This leads to a wastage of bandwidth, increases user's data consumption and battery drainage [11].

Table 2: Domains Accessed (2048 Gameplay)

Row Labels	**Requests per domain**	**Sum of Byte Count**	**Sum of Packet Count**
ads.get.it	3	131530	264
ads.mopub.com	6	5899	32
android.clients.google.com	2	8088	31
cpp-test.imp.mpx.mopub.com	3	3773	22
get.it	6	5607	51
in.metamx.com	3	3697	21
play.googleapis.com	3	384	6
rtb-05.get.it	5	4966	41
Grand Total	**31**	**163944**	**468**

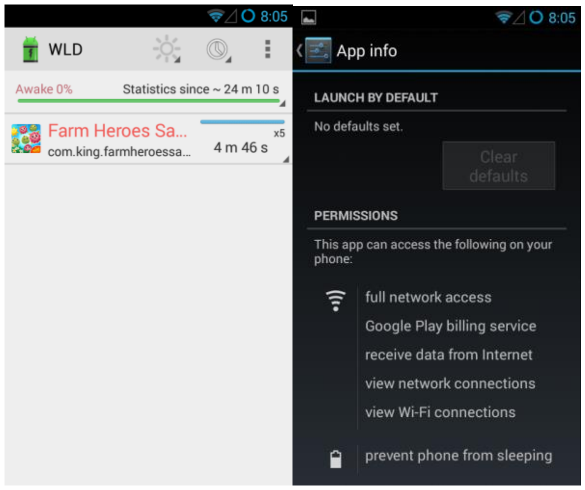
Recommendation: Some heuristic should be applied to detect when the running instance of the game has ended and only then requests for ads should be placed.

4.5 Wakelocks

Wakelock is a mechanism of power management service in Android OS, which can be used to keep CPU awake (Partial wakelock) and keep the screen on (Full wakelock). Applications which use wakelock privilege improperly lead to situations where the CPU or phone screen stays active without user's knowledge and thus causing battery drain [13].

Wakelocks on Android platform are used to ensure that the CPU does not sleep during some critical phase. But they also give the developer of a game the ability to prevent the CPU/screen from going to sleep. A typical smartphone app is written as a set of event handlers where events can be user or external activities. A developer of smartphone apps needs to keep a track of all possible scenarios of these events and handle wakelocks accordingly [10].

To detect wake locks in our chosen games, we used a freely available app on Google Play – Wakelock Detector, which shows wakelock usage statistics for all running apps. We found that Farm Saga misuses the wakelock, thus, keeping the screen awake, till the application is manually closed or put in the background. Figure 4a shows that within a span of just 4min 46sec Farm Saga triggered 5 wakelocks. This leads to battery wastage as the screen stays on unnecessarily for the entire duration of the game, even though the user is not interacting with the screen. In fact the game asks for user permissions to prevent the phone from sleeping (figure 4b), however most of the users unknowingly install the game.


8:05

WLD

Awake 0% Statistics since ~ 24 m 10 s

Farm Heroes Sa... x5
com.king.farmheroessa... 4 m 46 s

8:05

App info

LAUNCH BY DEFAULT

No defaults set.

Clear defaults

PERMISSIONS

This app can access the following on your phone:

full network access
Google Play billing service
receive data from Internet
view network connections
view Wi-Fi connections

prevent phone from sleeping

Figure 4: (a) Screen Wakelock in Farm Saga, (b) Required permissions for Farm Saga

Recommendation: Wakelocks should not be misused. The game should detect when there is no user interaction and then the game should allow the screen to turn off after the default timeout duration.

4.6 User Privacy

We observed that all the network traffic to the game server is using HTTPS connections, thus signing into Facebook or Google Play services to save game progress and achievements does not affect user privacy.

5. DISCUSSION

As future work, we would like to go through the implementation of the game, to identify issues like wakelocks, inefficient code, improper exception handling, multiple database calls, etc. We can also study other type of games, not covered in the current case study. These games include online multiplayer games like – *Clash of Clans* which requires continuous data transfer across network and high CPU and GPU intensive action games like *Counter Strike*.

We can study the benefits of offloading computation heavy code to a remote server to improve game performance and efficiency [7]. The current study focuses only on Android devices. This can be broadened to include iOS devices as well. We can implement the recommendations made in this paper and study the performance improvement.

In the current case study, we focus mainly on battery usage, network usage and energy efficiency. Future work might involve studying other resources. Also, the tests for game play with LTE were carried out on out personal phone. Though we used Greenify app to force close other apps, it does not close all the applications as the phone was not rooted. Hence there may be some services running in background which could not be stopped. Thus the recorded data usage may not be strictly due to game statistics. The current study is limited to the study of only two games. We are confident that studying more games belonging to different genres will enrich and add to the findings we have made so far. One of the major limitations we faced is that the game traffic is on HTTPS. Hence, we are unable to analyze the individual requests and responses of the data exchanged by the phone and the game server.

6. CONCLUSION

From the industry perspective, our project will help mobile app developers to improve their games and consume smartphone resources efficiently thus increasing games usage stats and revenues. From a consumer point of view, users will play games and still not require charging their phones often thus making better use of the mobile charge and the data plans. We show how following simple protocols like minifying the JSON and Java scripts, closing connections promptly and cache control mechanisms can help in efficient resource utilization in android games.

7. ACKNOWLEDGEMENT

We would like to thank Dr. Prof. Feng Qian for his guidance and help throughout the duration of this project. We appreciate the valuable advice given and the study material provided to us, without which this study would not have been a success. We also express our gratitude for all the help extended by Dr. Qian regarding the use of ARO.

8. Individual Contribution

Renuka worked on conducting the tests for Farm Saga in Idle and WiFi mode. She also conducted the tests for 2048 in LTE mode. Also, Renuka work on identifying the challenges, limitations and future work for this study. Renuka worked on the Discussion, Introduction, Methodology, Evaluation and Conclusion section of the paper. Milind conducted the test on Farm Saga LTE, 2048 Idle and 2048 WiFi. Milind worked on research for the project to identify study material for state-of-the-art and related work. Milind also worked on the Abstract, Related Study, Methodology, Evaluation and Conclusion section of the paper.

9. REFERENCES

[1] COMSCORE, "Games Account for 16% of the Total Time Spent on Mobile Apps in the US." Www.factbrowser.com. September 3, 2014. Accessed February 14, 2015. http://www.factbrowser.com/facts/15662/.

[2] Grubb, Jeff. "Here's What Candy Crush Saga and Kim Kardashian Are Doing to Your Phone's Storage and Battery Life." Www.venturebeat.com. August 26, 2014. Accessed February 14, 2015. http://venturebeat.com/2014/08/26/heres-what-candy-crush-saga-and-kim-kardashian-are-doing-to-your-phones-storage-and-battery-life/.

[3] JUNIPER RESEARCH, "Mobile Game Revenue Will reach $28.9B by 2016, up 38% from 2014." Www.factbrowser.com. June 18, 2014. Accessed February 14, 2015. http://www.factbrowser.com/facts/15109/.

[4] "High Risk Android™ Apps." High Risk Android Apps | Verizon Wireless. Accessed February 14, 2015. http://www.verizonwireless.com/support/high-risk-android-apps/.

[5] Zhang, Li, Chao Xu, Parth H. Pathak, and Prasant Mohapatra. "Characterizing Instant Messaging Apps on Smartphones."

[6] Schroeder, Stan. "Mobile Games Dominate Smartphone App Usage [STATS]." Accessed February 14, 2015. http://mashable.com/2011/07/07/smartphone-mobile-games/.

[7] Cuervo, Eduardo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl. "MAUI: making smartphones last longer with code offload." In Proceedings of the 8th international conference on Mobile systems, applications, and services, pp. 49-62. ACM, 2010.

[8] "AT&T Application Resource Optimizer (ARO)." AT&T Application Resource Optimizer (ARO). Accessed April 15, 2015.

[9] "Android Apps on Google Play." Usemon (Cpu Usage Monitor). Accessed April 15, 2015.

[10] Pathak, Abhinav, Abhilash Jindal, Y. Charlie Hu, and Samuel P. Midkiff. "What is keeping my phone awake?: characterizing and detecting no-sleep energy bugs in smartphone apps." In Proceedings of the 10th international conference on Mobile systems, applications, and services, pp. 267-280. ACM, 2012.

[11] Crussell, Jonathan, Ryan Stevens, and Hao Chen. "MAdFraud: investigating ad fraud in android applications." In Proceedings of the 12th annual international conference on Mobile systems, applications, and services, pp. 123-134. ACM, 2014.

[12] "ARO Best Practice." Text File Compression. December 11, 2014. Accessed May 2, 2015. http://developer.att.com/application-resource-optimizer/docs/best-practices/text-file-compression.

[13] "Android." Wakelock Detector-Save Battery. April 21, 2014. Accessed March 22, 2015. https://play.google.com/store/apps/details?id=com.uzumapps.wakelockdetector&hl=en.

[14] "ARO Best Practice." Multiple Simultaneous TCP Connections. December 11, 2014. Accessed May 2, 2015. http://developer.att.com/application-resource-optimizer/docs/best-practices/multiple-simultaneous-tcp-connections.

[15] "ARO Best Practice." Closing Connections. December 11, 2014. Accessed May 2, 2015. http://developer.att.com/application-resource-optimizer/docs/best-practices/closing-connections.

[16] "ARO Best Practice." Duplicate Content. December 11, 2014. Accessed May 2, 2015. http://developer.att.com/application-resource-optimizer/docs/best-practices/duplicate-content.

[17] "ARO Best Practice." Cache Control. December 11, 2014. Accessed May 2, 2015. http://developer.att.com/application-resource-optimizer/docs/best-practices/cache-control.

[18] Qian, Feng, Zhaoguang Wang, Alexandre Gerber, Zhuoqing Mao, Subhabrata Sen, and Oliver Spatscheck. "Profiling resource usage for mobile applications: a cross-layer approach." In Proceedings of the 9th international conference on Mobile systems, applications, and services, pp. 321-334. ACM, 2011.