Standby Energy Analysis and Optimization for Smartphones

Chengke Wang, Yao Guo, Yunnan Xu, Peng Shen, Xiangqun Chen
Key Laboratory of High-Confidence Software Technologies (Ministry of Education)
School of EECS, Peking University, Beijing, China
{wangchk11, yaoguo, xuyn14, shenpeng13, cherry}@sei.pku.edu.cn

***Abstract*—As smartphones become more and more powerful and complex, many research works have focused on the analysis and optimization of smartphone energy consumption. Most works are focused on the cases when smartphones are actively used. However, one major issue with smartphones is that the standby time has become much shorter compared with traditional feature phones. Many users have to recharge their phones everyday even though they are not using the phones very often. In order to understand what happens with smartphone battery while the phone is not in use, we combine current measurements with user trace analysis to reveal how smartphones consume energy in the standby mode. We have identified several factors that cause energy wastes in the standby mode. Then we propose a series of optimization methods to demonstrate the potential of energy reduction on smartphone standby energy. Results based on user traces show that we can extend the standby time by as much as 87% with the proposed optimizations.**

***Keywords*-Smartphones, energy consumption, standby energy, energy optimization, Android**

I. Introduction

Nowadays, smartphones are becoming more and more popular. Compared with traditional feature phones, smartphones are equipped with more powerful hardware components. Applications on smartphones are also more and more complex and powerful. However, the battery life of smartphones is reduced to as short as several hours for many heavy users. As a result, many users have to charge their smartphones every day.

To understand the energy consumption of smartphones, many research works have made efforts to analyze and optimize energy consumption of smartphones. Based on our observations, besides the energy consuming, heavy functions, another big issue with smartphones is that the standby time has become shorter and shorter, with many mainstream phones requiring recharges every day even for the light users.

While the advertised standby time for most smartphones are still in the range of several hundred hours, our experiences with different types of smartphones show that it is absolutely impossible to reach even close to a hundred hours.

Our previous work on energy estimation and evaluation on a large number of smartphones showed that many users spent more battery in the standby mode than when the phone is running apps in the foreground [22]. The average current in the standby mode reaches up to 50mA, which is one magnitude higher than the measured current of around 5mA in real standby mode.[1] Moreover, our statistical results show that standby mode accounts for 81% of the total using time and 58% of the total battery capacity.

What happens when the phone is not in use? How is the battery consumed? What causes the energy surges? Is there any way to optimize the standby power? These are the questions we tried to answer in this paper.

In order to perform energy analysis in the standby mode, we first conduct a series of hardware measurements. According to our experimental results, the battery consumption of a smartphone while not in use is dependent on the actual context. We have observed the following factors that affect standby energy consumption in various contexts:

- When we switch off the screen, the phone takes several seconds to reach a stable low-power mode.
- More energy is consumed when we switch the smartphone apps to the background instead of terminating them explicitly.
- More energy is consumed when many hardware components are not turned off, such as Wi-Fi.
- Many background services wake up the phone periodically in the background, causing frequent power surges.

Some of the factors are also significant consumers in non-standby mode. However, we want to focus on how they consume the standby energy. To quantify the effects of these context factors on the standby energy, we have also developed a lightweight tool to collect event traces consisting of important system events on smartphones of 9 volunteers. After analyzing the user traces, we are able to obtain the following results:

- In the user traces, 85% of the total CPU time, 16.7% of the Wi-Fi traffic and 2.8% of the cellular traffic are consumed in the standby mode. Thus, the majority of CPU resource is consumed in the standby mode, while the majority of network resource is consumed in the non-standby mode.
- In all types of resource usages in the standby mode, system services use the most CPU, online and downloading apps use the most of Wi-Fi network, while chat

[1]For a smartphone equipped with a standard 1500mAH battery, 5mA standby power means 300 hours standby, while 50mA means the standby time is roughly 30 hours.

apps and SNS apps use the most of cellular network.

Based on our analysis and estimation, with all these activities, standby energy has been dominated by these extra energy consumption in many cases, suggesting potential opportunities for optimization. We propose several methods to optimize the standby energy:

- *Apply fast dormancy to reduce tailing energy spent after screen-off.* We can apply fast dormancy scheme (similar to the techniques for the 3G network) to the screen off tails to shorten the tail time.
- *Turn off unused hardware components* (e.g. Wi-Fi module). According to our measurements, Wi-Fi connection with no traffic is also a big consumer of standby energy. We can switch off Wi-Fi in the standby sessions that do not need it.
- *Delay network packages to group them.* Small traffic wastes a lot of energy in the standby mode due to the tail energy of network transmission. We can delay some of the packages to group them together to reduce the tail energy.
- Automatically *terminate* or notify the users to terminate the apps that consume significant energy in the background.

We apply these methods to our measurement data and usage traces, the result shows that we can potentially extend the standby time by as much as 87.3% on a typical smartphone.

The main contributions of our work are as following:

- We have performed hardware measurements of current traces of a standby smartphone in various contexts and identified the main factors that affect the standby energy. These factors include screen off tails, hardware status, background applications and services.
- We have performed a user study of 9 volunteers for about one week. According to the resource usage information of these users, we calculate and compare the main factors affecting standby energy.
- We have presented several optimization methods to extend the standby time of smartphones and estimate the effect of each approach based on our measurement data and user traces. The result shows that we can extend the standby time by as much as 87.3% with these methods.

II. Methodology

In order to understand what happens when a smartphone is not in use, we have taken an approach combining hardware measurement with trace analysis.

A. Hardware Measurement

Hardware measurement is applied to measure the parameters of a smartphone via external hardware tools at runtime. Our hardware measurement includes power meter and network monitor.

1) Power Meter: We use the Monsoon Power Monitor [12] in our study to measure the current value in various stages. The Power Monitor also supplies a stable voltage to the smartphone. Since the voltage is a constant, we simply use the current value to denote the power consumption.

Due to the huge amount of data generated by the Power Monitor, we are unable to perform analysis on a long trace such as several hours. Instead, based on our analysis on traces, we selectively monitor the current traces for short periods on various representative contexts.

With the current traces, we have identified some power consumption patterns when the smartphone is in the standby mode. Furthermore, when we compare the power patterns with the event traces collected on user smartphones, we are able to determine the causes of these power patterns.

2) Wi-Fi Traffic Monitor: During hardware measurement experiments, we have not plugged-in a SIM card into the smartphone. Thus all the network traffic is through Wi-Fi.

Our approach for monitoring the Wi-Fi traffic is to set up a Wi-Fi AP on a laptop, then connect the smartphone to this AP. When the smartphone sends or receives a network package, we monitor the packages on the laptop using the WinPcap [21] tool.

With Wi-Fi traffic traces and power traces, we can identify the power consumption caused by Wi-Fi traffic in the current traces. We are then able to analyze network-related power consumption in different contexts.

B. User Trace Collection

We developed a light-weight tool to collect event traces and resource usage information on actual smartphones used by some volunteers. The tool consists of two parts:

- An event-driven collector that records the event traces. When the mobile operating system broadcasts an event, our tool catches the occurrence of the event and its timestamp.
- A polling collector that reads the information on resource usage such as CPU and network. This part reads CPU time and network traffic of each running app and service every 10 seconds.
  In order to determine the polling frequencies, we have tried various frequencies. We conclude that a time interval of 10 seconds could be accurate enough for our following analysis, and the overhead is tolerable according to the feedback of our volunteers.

Please note that the user traces collected with this tool are only used to record user behaviours. The energy values are calculated based on our hardware measurement. Thus the overhead of the data collection tool would not impact our results.

1) Event Trace: Our tool runs as a background service on the smartphone and listens to the following events:

- Screen events: Screen is turned on/off. We denote standby mode as the time periods with the screen

of smartphones off. With the screen events, we can distinguish the standby periods.
- Network events: including events when network is enabled/disabled, or network state is changed.

With these events, we can divide the smartphone usage into many sessions. Each session runs in a relatively stable context.

2) Resource Usage: The background service of the tool also reads the CPU and network usages, and summarize the usage information for each session.

- CPU usage: CPU usage is denoted using CPU time for each running app and service.
- Network usage: We denote the network usage by the number of traffic bytes for each running app and service. Since we collect the trace when the user actually uses the phone, we also need to distinguish between different network environments for Wi-Fi or cellular (3G/4G).

We can read the CPU time and network transmission bytes of each app from the file in the "/proc" directory.

III. Power Trace Analysis

In this section, we introduce our experimental setup and present standby energy consumption patterns observed in different contexts.

A. Experimental Setup

We use a Google Nexus S smartphone to perform hardware measurement. The power numbers are measured with the Monsoon Power Monitor, which also supplies a stable 4.2 V voltage to the smartphone.

We install popular apps including Wechat, UCbrowser and AngryBirds on the phone and measure the power numbers in different contexts.

B. Observations from Current Measurement

Based on our experiments, we have identified several factors of standby energy consumption in different contexts.

1) Tail Energy during Screen Switch-Off: One interesting aspect in our observations is that each time when the screen is off, the smartphone will keep in a high power stage for several seconds even after the screen is off, before it enters the real standby mode with a low-level current.

Figure 1 shows the measured current when the system turns off the screen automatically after a specified period with no user operations. The screen is turned off at around 1.5s. It is obvious that the current increases at first. After about 10 seconds, the current falls to about 5 mA, which is the norm for a phone in standby. The result suggests that the power consumption of the smartphone usually could not fall down to the lowest level immediately (or real fast) after the screen is turned off. There exists "tail energy", which is similar to previous studies on network traffic. [1]

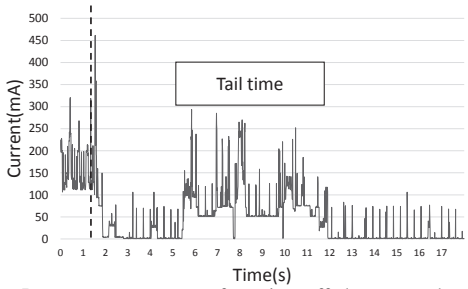

Figure 1. Power measurement of turning off the screen by the system (Wi-Fi on).

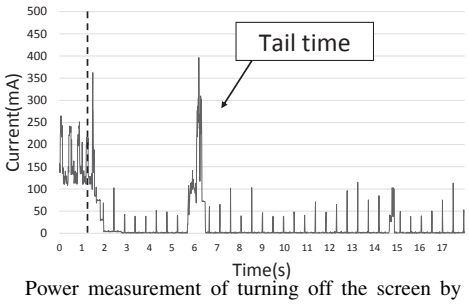

Figure 2. Power measurement of turning off the screen by the system (Wi-Fi off).

When measuring with Wi-Fi off, we observe a much shorter "tail time" in Figure 2. Compared with the tail time of about 6.6 seconds with Wi-Fi on, the tail time is only 0.7 seconds when Wi-Fi is off. The average current values of the two situations are almost the same, at about 82.02 mA. Thus the tail energy would be much less if Wi-Fi was disabled before the screen is off.

During these "tail times" above, we do not observe any Wi-Fi traffic of the smartphone on the notebook Wi-Fi AP. We are still investigating exactly what happened during this period. Our best guess is that the smartphone may be preparing for the standby mode, but it may keep some resources (such as Wi-Fi connection) alive for a while in case the user would turn on the screen in a short time.

According to this guess, the system may perform some optimization if the screen is turned off by the power key, which suggests the phone probably not used in a short time. Then we test this situation and the result is presented in Figure 3. It shows that there would be no significant "tail energy". Thus, the system do optimize the "tail energy" when it "assumes" the user might not use the phone for a while.

This suggests that if the system is able to detect user intentions, it could also apply the fast dormancy approach when system switches off screen due to inactivity.

2) Effect of Hardware Status: Another observation (which has been exploited by many energy management tools) is that the standby energy could be kept at a minimum in the flight mode, when all communication channels/devices are turned off.

In order to show how the status of hardware components

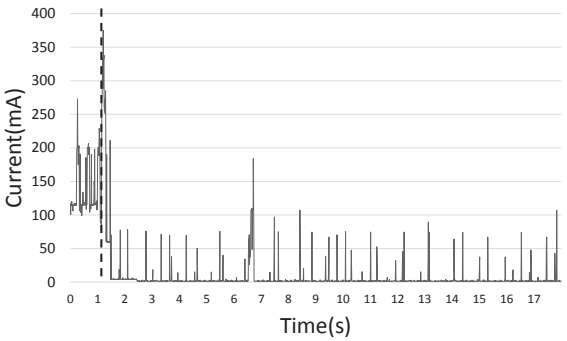

Figure 3. Power measurement of turning off the screen by the user (Wi-Fi on).

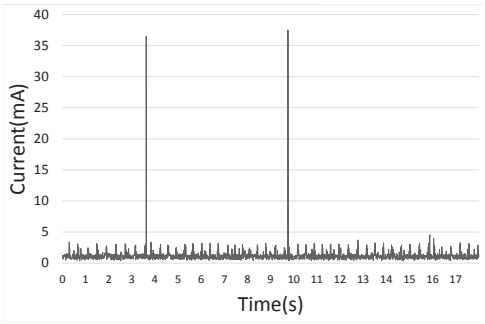

Figure 4. Power measurement of flight mode with no background applications.

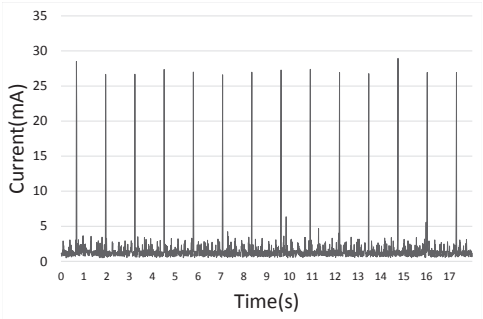

Figure 5. Power measurement of standby mode with Bluetooth on but no background applications.

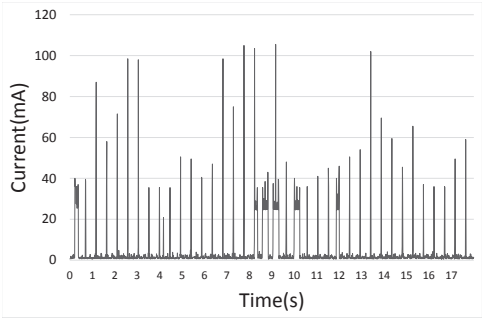

Figure 6. Power measurement of standby mode with Wi-Fi on but no background applications.

could affect the background power, we first measure the baseline current curve in the flight mode (with screen off and no background applications and services), as shown in Figure 4. The average current is only 1.15 mA.

Then we measured the current curves in the standby mode with Bluetooth and Wi-Fi enabled (but no traffic), shown in Figure 5 and 6 respectively. Although when Bluetooth is enabled, it shows only small energy waves, little extra energy increase. We see that when Wi-Fi is enabled, the base current is increased to 1.39 mA, while the average current value is 5.72 mA.

The results shows that the hardware components would consume significant energy in the standby mode even they are not used. We should turn off them to save the battery if not using them for a long period of time.

3) Effect of Background Applications: When turning to a new task, many smartphone users may simply switch the old app to the background instead of terminating it (especially for iPhone/iPad apps). For example, when the user receives a phone call while playing a game, he might simply switch the game to the background and may not use it again in a long time.

Do the applications remaining in the background cost energy? Figure 7 shows the current curve in the flight mode with many applications running in the background. In most of the time, the current value is the close to the baseline current in Figure 4. But occasionally, it may jump to a high value of about 90 mA for a short period. Since the phone is in the flight mode in this test, the energy can not be consumed by network traffic. It may be related to the background applications and system scheduling.

4) Effect of Background Services: Even with no background applications, many smartphones also run multiple services in the standby mode.

Do these services consume the battery? Figure 8 shows the current curve in the standby mode with Wi-Fi enabled. We can see the current curve periodically rises up for about every 2 seconds. We find that each time the current rises, one or more Wi-Fi packages would occur on our notebook AP. It suggests that the energy is related to the Wi-Fi traffic. We test the Wi-Fi traffic for about 50 times. The average time of a single current pattern of Wi-Fi traffic is about 1.98 seconds with an average energy consumption of about 53.82 uAh.

Are there any other components consume the energy in these current patterns? Figure 9 shows the current curve when we perform a similar experiment without Wi-Fi. The current patterns disappear and the average current decreases to about 5.31 mA. Thus, the Wi-Fi traffic consumes almost all the energy in these current patterns.

The results suggest that many applications would run services on the smartphone. Some of these services always produce network traffic periodically in the standby mode if network is available. We believe this is one of major contributors to standby energy that decreases smartphone standby time significantly.

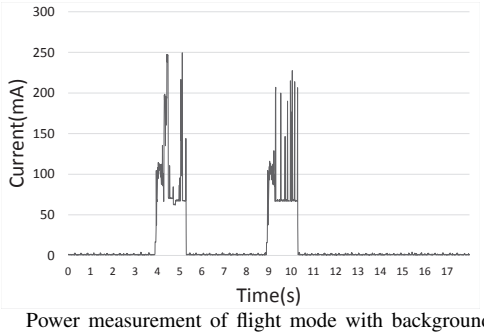

Figure 7. Power measurement of flight mode with background applications.

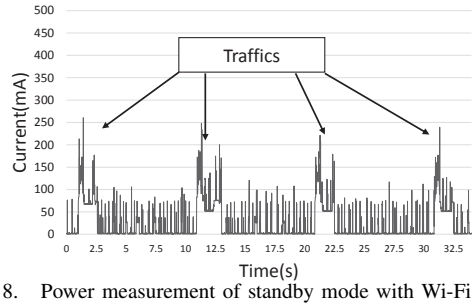

Figure 8. Power measurement of standby mode with Wi-Fi enabled.

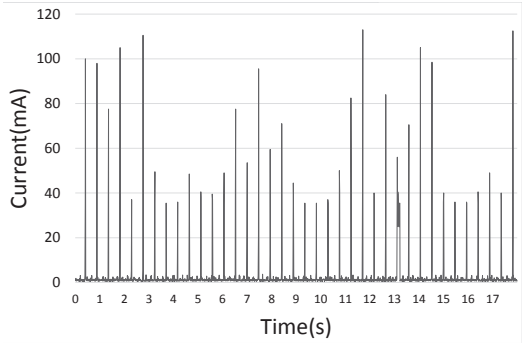

Figure 9. The current of standby mode without Wi-Fi.

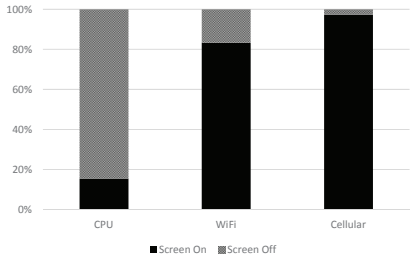

Figure 10. Resource usage of apps: screen-off vs. screen on.

IV. User Trace Analysis

In order to study the standby energy in real usage scenarios, we collected user traces from 9 volunteers for about one week.

A. Data Collection

We collected the user traces when the volunteers used their phones normally. All the users have installed our collection tool on their smartphones. Once they click the "start" button in our tool, the background service starts and it restarts automatically when the smartphones are powered on. The tool records the traces and resource usage information, then writes it to a data file. After a week, we asked the volunteers to submit their data files.

B. Smartphone Usage in User Traces

We first study the basic information of smartphone usage of these users, as shown in Table I.

We denote each user with a number from 1 to 9. Among different users, the number of apps ranges from 46 to 142, which includes the actual apps, services of apps, and also system services. The average trace time of all users is about 86 hours during the week, since we would not collect any data when the smartphones are powered off.[2] The light users use their smartphones only for 9% of the time, while heavy users use their phones for 30% of the time. As a results, even the smartphones of heavy users have spent more time in the standby mode than in non-standby mode.

[2]The trace time of User 5 is abnormal. Because of our collection tool has a bug, it could not always start automatically when the phone of User 5 is powered on.

C. Application Usage in User Traces

The root causes of standby energy are software behaviors. Besides the five parts of standby energy we have observed and analyzed, we also want to know how the background apps and services behave in the standby mode and which apps consume the most energy.

1) Application Behaviors: Standby vs. Non-standby: As we know, main energy consumers of smartphones are CPU, network and screen. In the standby mode, during which the screen is turned off, the main energy consuming components would be CPU and network.

We add up the CPU and network usage of all the apps in the standby/non-standby mode. The result is shown in Figure 10.

Compared with non-standby mode, more CPU time is consumed in the standby mode. The main reason is that the standby mode accounts for 87% of all the trace time. The average CPU utilization in the standby mode is 23.9%, while the number is 29.8% in the non-standby mode. It suggests that, the majority of CPU time is consumed in the standby mode, but the consumption rate is smaller than that in the non-standby mode. Wi-Fi and cellular network traffic in the standby mode is much less than in the non-standby mode.

2) Major Applications of Standby Energy Consumer: Table II shows the top apps and services consuming most of CPU, Wi-Fi and cellular network in the standby mode.

We can see among the top ten CPU consumers, six of them are system services including phone calls, system media and system UI. The top Wi-Fi consumers are network apps, for example the Wandoujia app store, Youdao online dictionary and Huawei network drive. The top cellular net-

Table I
USER TRACE STATISTICS.

User ID	Total Trace Time(h)	Standby Time(%)	Using Time(%)	App Number
1	137.9	85.2%	14.8%	142
2	96.8	83.8%	16.2%	47
3	126.1	91.0%	9.0%	140
4	62.1	90.5%	9.5%	112
5	3.6	69.9%	30.1%	48
6	112.7	86.9%	13.1%	99
7	92.2	91.6%	8.4%	99
8	45.3	79.2%	20.8%	76
9	104.3	87.0%	13.0%	46

Table II
TOP APPS AND SERVICES REGARDING THEIR RESOURCE USAGES.

Top CPU consumers	Top Wi-Fi consumers	Top Cellular Consumers
com.android.phone	com.wandoujia.phoenix2	com.android.email
android.process.acore	info.matthewwardrop.scholarley	cn.com.fetion:service
android.process.media	com.youdao.dict	cn.com.fetion
com.android.systemui	com.youdao.dict:yddictclip	com.tencent.mobileqq:MSF
com.taobao.taobao	android.process.media	com.sina.weibo
com.taobao.taobao:pushservice	com.estrongs.android.pop	com.sina.weibo.servant
com.android.mms	cn.wiz.note	com.tencent.mobileqq
com.android.email	com.huawei.hidisk	com.tencent.mm:push
com.baidu.appsearch:bdservice_v1	com.ting.mp3.android	com.evernote.world
com.baidu.BaiduMap:bdservice_v1	com.ideashower.readitlater.pro	com.xiaomi.xmsf

work consumers are SNS apps and chat apps, such as Fetion, QQ and Weibo. It suggests that the top consuming apps of different resources in the standby mode is also different. When optimizing the standby energy of each resource, we should mainly focus on the corresponding groups of apps that consume more energy.

D. Energy Usage in User Traces

In Section III we present several factors of standby energy including hardware causes and software causes. With the user traces of smartphone usage collected from users, we can compare the factors of standby energy consumption.

1) Calculating Energy Usage with User Traces: Since the user devices are different, we just take some statistical results of smartphones usage from user traces and adapt them to our hardware measurement numbers of Nexus S to calculate the approximate energy consumption. For example:

- *Screen-off tai*l: We denote the energy consumption for one screen-off tail by the average energy of the "tail time" according to our measurement on Nexus S, while the frequency of screen-off is counted from the screen-off events in our user traces.
- *Network traffic*: We model the traffic energy based on traffic bytes with our measurement results on Nexus S, while the total traffic bytes is counted from the network usage information of our user traces.

2) Energy Distribution in the Standby Mode: According to our observations of standby energy in Section III-B, we divide the standby energy into five parts:

- Flight mode base power: As we have measured, flight mode with no running apps can be used as the baseline of the standby power.
- Screen off tail: Another part is the energy consumed during the "tail time" after the screen is turned off.
- Background apps: The apps running in the background can cause occasional rises of standby energy.
- Wi-Fi connection: Even though with no traffic, only the Wi-Fi connection consumes extra energy.
- Wi-Fi traffic: When background services communicate with servers or other devices, the Wi-Fi traffic consumes energy.

Combining the user traces with power measurements, we are able to calculate these components to form a big picture of the actual standby energy.

According to Figure 4, the baseline current in the flight mode is 1.15 mA. When Wi-Fi is enabled, the average current is increased by 4.67 mA, as Figure 6 shows. This corresponds to the Wi-Fi connection power. According to Figure 7, if several apps are running in the background, the current would occasionally rises up to about 90 mA and last about 1.1 second. We have not found the reason or a pattern. However, we tested this context for 10 minutes and the current rises up for 24 times. Thus we roughly regard it as rising up every 25 seconds. It is equivalent to a 3.96mA current throughout the standby time.

To calculate the screen off tail energy, we count the frequency of screen off events in our trace data. In the traces of all the 9 users, the number of screen off events is 1,987 during the 681.2 hour total using time. Thus the

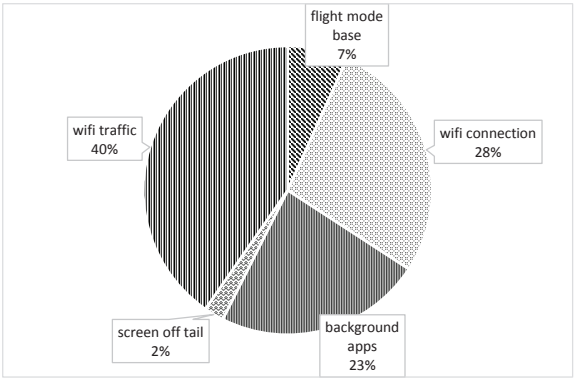

Figure 11. The distribution of standby energy.

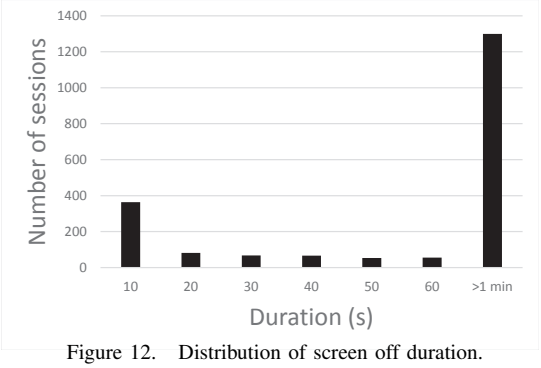

Figure 12. Distribution of screen off duration.

frequency is turning off the screen every 0.39 hours. We do not distinguish whether the screen is turned off by the user or by the system. If all the screen-off events were caused by the system with Wi-Fi enabled, then each screen-off event would have a 6.6-second "tail time" with 82.02 mA average current. Thus, the screen tail is equivalent to a 0.38mA current throughout the standby time.

According to Figure 8, the average energy consumption of one traffic pattern is 53.82 uAh and the average time is 1.98 seconds. The average traffic bytes is 6.4K bytes as we monitored on our notebook AP. Thus we modeled the average energy of 1K byte traffic at 8.35 uAh. In our user traces, the standby Wi-Fi traffic is 547 MB during 681.1 hours. The average is 233 bytes per second, which consumed 1.9 uAh according to our model. On average, the traffic power is equivalent to a 6.87mA current throughout the standby time.

We have summarized the calculation results and shown the distribution of standby energy in Figure 11.

From the distribution of standby energy, we observed the following conclusions:

- The baseline energy in the flight mode is a very small part of the total standby energy.
- Wi-Fi network (Wi-Fi connection and Wi-Fi traffic) consumes the most energy in the standby mode with a percentage of about 70%. However, the number of standby Wi-Fi traffic bytes only accounts for 16.7% of all the Wi-Fi traffic. It suggests that the network energy in the standby mode has a large optimization space.
- The screen off tail energy seems to consume a very significant energy of 150.4 uAh per tail. However, this part accounts for a very small percentage in the total standby energy. Since the standby time is a very long period, persistent energy consumption such as Wi-Fi connection and periodically Wi-Fi traffic would be much more significant.
- Background apps also consume a significant part of standby energy. Thus, terminating unused apps explicitly can help extending the standby time.

V. Standby Energy Optimization

In this section, we propose several optimization methods to extend the standby time based on our analysis above, and show the effectiveness of these methods based on our user traces.

A. Fast Dormancy

In Section III-B1, we observe the tail energy after the screen is turned off. We can also see that if users turn off the screen manually, the system would apply a mechanism similar to the fast dormancy of 3G network to reduce the "tail energy". Thus, a straightforward method to optimize the tail energy is to apply the mechanism to more screen-off sessions that the users were not going to turn on the screen in a short period.

We analyze the screen off sessions in our user traces and show the duration distribution of screen off sessions in Figure 12.

The result shows that the number of screen off sessions that last more than 1 minute is 1,299 out of the total number 1,987. If we apply fast dormancy to these sessions, we can save about 65% of the screen off tail energy.

According to our calculation in Section IV-D2, the "tail energy" accounts for about 2% of the standby energy, thus "fast dormancy" could save 1.3% of the total standby energy.

B. Turn off Unused Hardware Components

As the results in Section III-B2, the hardware components would consume energy when they are even not used. For example, the Wi-Fi module consumes about 28% of the standby energy even when no bytes are sent or received via Wi-Fi. It is a significant part of the standby energy.

According to our user study in Section IV-C, we know that the traffic in the standby mode is much less than in the non-standby mode. The main network consumers in the standby mode is network apps such as app stores, online notes, SNS apps and chat apps. Network usage of network apps usually happens if the screen was turned off while the apps are downloading files or updating data. After the ongoing traffic of downloading or updating is over, these network apps would produce little network traffic. The SNS apps and

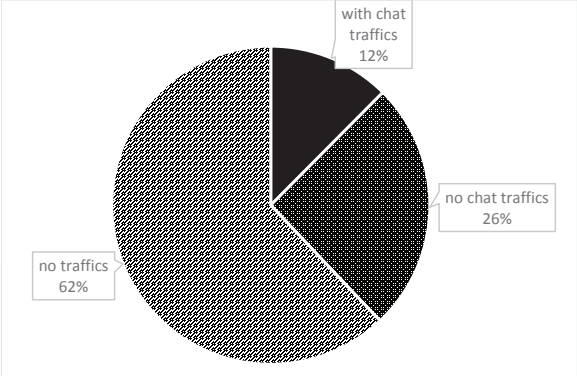

Figure 13. Traffic distribution of standby sessions.

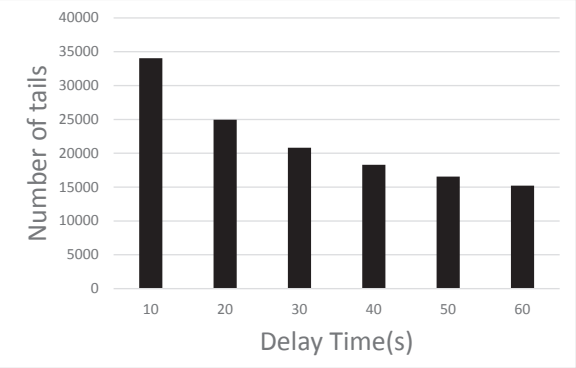

Figure 14. Number of network tails with different delay time.

chat apps, however, always need to receive messages from their server. Thus they need a persistent network connection.

We can use different optimization strategies in different contexts. If none of the running apps require a long period network connection, we could simply disable the network module after all the ongoing traffic is over. Otherwise, we must hold the connection. However, we can still save energy with other methods in this situation. For example, we can turn to a more energy saving network type (e.g. disable Wi-Fi and turn to the 3G network.)

To estimate the effect of this method, we focus on the most popular chat apps and SNS apps among our volunteers: QQ, Weibo, Wechat and Fetion. We analyze the traffic of all the screen-off sessions. The result is shown in Figure 13. It shows that 1,381 sessions (422.7 hours) have no Wi-Fi traffic. It suggests that we can switch off Wi-Fi during these sessions which last for 422.7 hours out of the total 681.2 hours of standby. It could save about 14.3% of the total standby energy! Furthermore, if we switch off Wi-Fi in the 1,585 sessions (596.4 hours) with no traffic for chat apps, we could save about 20.1% of the total standby energy!

C. Group or delay the network traffic

In the results of Section III-B4, we can see that the network traffic in the standby mode is small and periodical. The traffic is mainly caused by the SNS apps or chat apps, because they have to communicate with their servers to check messages. The small traffic can cause a lot of tail energy for network accesses. On the other hand, users can tolerate a much larger latency in the standby mode than in the non-standby mode. Thus, we can find more opportunities to perform traffic grouping or delaying to reduce network tail energy.

According to the previous study of network tail energy [1], we can divide the current pattern of network traffic in Figure 8 into two periods. The first period, during which the packages are transferred, has a higher average current. The second period with a stable and lower current is the "tail time" of network accesses. We can see the tail time lasts about 1 seconds with a relatively stable current of 70 mA, consuming about 19.4 uAh energy.

In our user traces, we can find the tail time by network usage traces. We have recorded a total of 44,761 tails. Thus the tail energy accounts for 18.6% of the Wi-Fi traffic energy, therefor accounts for 7.4% of the standby energy.

If we delay some of the packages to group them together, then the tail time could be reduced. Figure 14 shows the number of tails if we group the traffic in the standby mode in every 10 seconds, 20 seconds, etc. For example, if we group the traffic in every 1 minute, the number of tails would decrease to 15,225. Thus, we could save 66% of the tail energy, which accounts for 4.9% of the total standby energy.

D. Terminate background applications

When leaving a foreground app, many smartphone users may simply switch the app to background, but forget to terminate it. According to our measurement data in Section III-B3, that could waste a lot of energy if the smartphone turns into the standby mode with these background apps running. To reduce the standby energy of background applications, we could provide some warning messages to users when they switch from the foreground application to system home page or other applications. In a more complicated measure, we can also try to predict user behaviors and habits to terminate background applications automatically.

We analyze our user data to find out how many apps running in standby mode could be terminated. We focus on the apps that run in a standby session, but are not used (i.e. switched to foreground) in the next non-standby session.[3] Figure 15 shows the distribution of these apps. The result suggests that about 80% of standby sessions runs more than 7 apps that are not used in the next session. If we terminate these apps, the time of background apps running in the standby session could be reduced by 760,251s. As the total time of background apps in the standby sessions is 860,815s,

[3]We notice that some apps and services have never run in the foreground, thus we only analyze the apps that might run in the foreground.

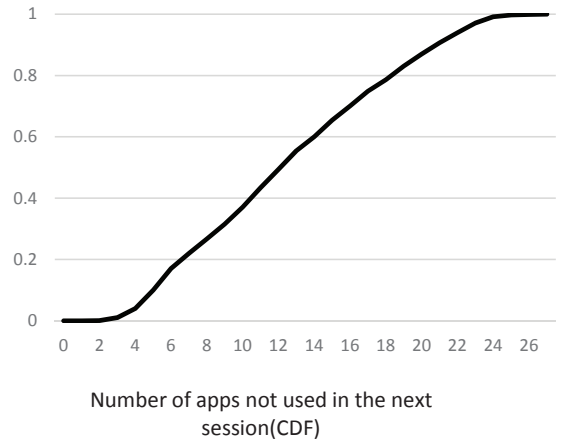

Figure 15. Distribution of background apps not used in the next session.

we could save 88.3% of the energy consumed by background apps, which accounts for 20.3% of the total standby energy.

E. Overall Optimization Results

If we combine these optimizations present above, we are able to save about 46.6% of the standby energy. Thus we can extend the standby time for a smartphone by as much as 87.3%.

Please note that although we use only rough calculations, it shows that these optimizations are promising. Furthermore, as the optimizations we have proposed are all straightforward techniques, there exists more potential to reduce standby energy and extend battery life if we investigate the issue more extensively and apply more complicated context-aware optimizations.

VI. Related Works

Smartphone energy is a hot topic in recent years. Many research works have been focus on analyzing the energy consumption on smartphones.

At the level of hardware components, Dong *et. al* [5] modeled the power consumption of OLED screen by the color (average RGB value) of the display content. CABLI [26] modeled the quantitative relation between system context attributes and the battery discharge rate based on multiple linear regressions. TailEnder [1] developed a model for the energy consumed by network activity for each network technology. They also presented a method to reduce the tail energy of the smartphone network. Li *et. al.* [15] analyzed the energy consumption of storage stacks of mobile devices. At the software level, Eprof[19] modeled the power consumption of smartphone system by system call trace. Appscope[24] modeled the power consumption of smartphone apps by their resource usage. eLens[9] estimated the energy consumption of source code lines. Lee *et. al.* [13] looked into the applications to analyze the energy consumption of each UI states. Dong *et. al.* [6] proposed a method to calculate the energy of each app in the system which several apps are running at the same time.

Many other works make efforts to optimizing the power consumption. For example, E-MiLi [25] reduces the power consumption in idle listening by sleep scheduling in the time spent in idle listening, Catnap [4] optimizes energy consumption of mobile devices by allowing them to sleep during data transfers. Deng *et. al.* [3] uses a traffic-aware technique to reduce the tail energy of network traffic by impact the network traffic and fast dormancy. Hsiu *et. al.* [10] reduces the energy of video player by turning down the backlight. Li *et. al.* [14] optimizes the display energy of web apps on smartphone by change their color scheme.

User study is also a commonly used approach to analyze smartphone usage and energy consumption. Falaki *et. al* [7] studied 255 smartphone users, characterized user activities and applications, and the impact of those activities on network and energy usage. Ferreira *et. al* [8] presented a 4-week study of more than 4,000 people to assess their smartphone charging habits to identify power intensive operations. Oliver *et. al* [18] conducted one of the largest-scale study to measure the energy consumption of 20,100 BlackBerry smartphone users, and predicted energy level within 72% accuracy in advance. Carat *et. al* [17] proposed a collaborative method to diagnose abnormal energy drains of smartphone applications with about 400,000 users.

Some research works have studied background energy. For example, Xu *et. al.* [23] optimized the energy consumption of email sync in the standby mode. Huang *et. al.* [11] analyzed the screen-off network traffic. Pathak *et. al.* [20] found a category of energy bugs of mobile apps which keeps the CPU or screen awake abnormally. Bolla *et. al.* optimized the energy of background apps by an Application State Proxy which suppresses/stops the applications on smartphones and maintains their presence on other network device. Martins *et. al.* [16] optimized background energy by instrumented Android system to interpose on signals that cause task wakeups. Chen *et. al.* [2] analyzed the energy of background apps. However, no existing work has attempted to provide a detailed analysis on what causes the energy wastes while the phone is not in use.

VII. Concluding Remarks

Understanding the energy consumption of smartphones is very important for both users and application developers. This paper presents an analysis on standby energy of smartphones through power measurement and collected user traces. We have identified several main energy consumers in the standby mode and proposed several methods to optimize the standby energy. The results show that we can extend the standby time by more than 87.3% with these methods.

Although our study is based on estimation and calculations, the observations from our experiments reveal many aspects of the standby energy consumption of smartphones.

We believe standby energy consumption is an important issue and investigating the issue further can help many smartphone users, developers and researchers.

ACKNOWLEDGMENT

This work is supported in part by the High-Tech Research and Development Program of China under Grant No. 2015AA01A202, and the National Natural Science Foundation of China under Grant No. 61421091, 61103026.

REFERENCES

[1] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. Energy consumption in mobile phones: A measurement study and implications for network applications. In *Proceedings of IMC '09*, 2009.

[2] X. Chen, A. Jindal, N. Ding, Y. C. Hu, M. Gupta, and R. Vannithamby. Smartphone background activities in the wild: Origin, energy drain, and optimization. In *Proceedings of MobiCom '15*, pages 40–52, New York, NY, USA, 2015. ACM.

[3] S. Deng and H. Balakrishnan. Traffic-aware techniques to reduce 3g/lte wireless energy consumption. In *Proceedings of CoNEXT '12*, 2012.

[4] F. R. Dogar, P. Steenkiste, and K. Papagiannaki. Catnap: Exploiting high bandwidth wireless interfaces to save energy for mobile devices. In *Proceedings of MobiSys '10*, 2010.

[5] M. Dong, Y.-S. K. Choi, and L. Zhong. Power modeling of graphical user interfaces on oled displays. In *Proceedings of DAC '09*, 2009.

[6] M. Dong, T. Lan, and L. Zhong. Rethink energy accounting with cooperative game theory. In *Proceedings of MobiCom '14*, pages 531–542, New York, NY, USA, 2014. ACM.

[7] H. Falaki, R. Mahajan, S. Kandula, D. Lymberopoulos, R. Govindan, and D. Estrin. Diversity in smartphone usage. In *Proceedings of MobiSys '10*, 2010.

[8] D. Ferreira, A. K. Dey, and V. Kostakos. Understanding human-smartphone concerns: a study of battery life. In *Proceedings of Pervasive'11*, 2011.

[9] S. Hao, D. Li, W. G. J. Halfond, and R. Govindan. Estimating mobile application energy consumption using program analysis. In *Proceedings of ICSE '13*, 2013.

[10] P.-C. Hsiu, C.-H. Lin, and C.-K. Hsieh. Dynamic backlight scaling optimization for mobile streaming applications. In *Proceedings of ISLPED '11*, 2011.

[11] J. Huang, F. Qian, Z. M. Mao, S. Sen, and O. Spatscheck. Screen-off traffic characterization and optimization in 3g/4g networks. In *Proceedings of IMC '12*, 2012.

[12] M. S. Inc. Monsoon power monitor. https://www.msoon.com/LabEquipment/PowerMonitor/.

[13] S. Lee, C. Yoon, and H. Cha. User interaction-based profiling system for android application tuning. In *Proceedings of UbiComp '14*, pages 289–299, New York, NY, USA, 2014. ACM.

[14] D. Li, A. H. Tran, and W. G. J. Halfond. Making web applications more energy efficient for oled smartphones. In *Proceedings of ICSE 2014*, pages 527–538, New York, NY, USA, 2014. ACM.

[15] J. Li, A. Badam, R. Chandra, S. Swanson, B. Worthington, and Q. Zhang. On the energy overhead of mobile storage systems. In *Proceedings of FAST'14*, pages 105–118, Berkeley, CA, USA, 2014. USENIX Association.

[16] M. Martins, J. Cappos, and R. Fonseca. Selectively taming background android apps to improve battery lifetime. In *Proceedings of USENIX ATC '15*, pages 563–575, Berkeley, CA, USA, 2015. USENIX Association.

[17] A. J. Oliner, A. P. Iyer, I. Stoica, E. Lagerspetz, and S. Tarkoma. Carat: Collaborative energy diagnosis for mobile devices. In *Proceedings of SenSys '13*, 2013.

[18] E. A. Oliver and S. Keshav. An empirical approach to smartphone energy level prediction. In *Proceedings of UbiComp '11*, 2011.

[19] A. Pathak, Y. C. Hu, M. Zhang, P. Bahl, and Y.-M. Wang. Fine-grained power modeling for smartphones using system call tracing. In *Proceedings of EuroSys '11*, 2011.

[20] A. Pathak, A. Jindal, Y. C. Hu, and S. P. Midkiff. What is keeping my phone awake?: Characterizing and detecting no-sleep energy bugs in smartphone apps. In *Proceedings of MobiSys '12*, pages 267–280, New York, NY, USA, 2012. ACM.

[21] R. Technology. Winpcap. http://www.winpcap.org/.

[22] C. Wang, F. Yan, Y. Guo, and X. Chen. Power estimation for mobile applications with profile-driven battery traces. In *Proceedings of ISLPED '13*, 2013.

[23] F. Xu, Y. Liu, T. Moscibroda, R. Chandra, L. Jin, Y. Zhang, and Q. Li. Optimizing background email sync on smartphones. In *Proceeding MobiSys '13*, 2013.

[24] C. Yoon, D. Kim, W. Jung, C. Kang, and H. Cha. Appscope: Application energy metering framework for android smartphones using kernel activity monitoring. In *Proceedings of USENIX ATC'12*, 2012.

[25] X. Zhang and K. G. Shin. E-mili: Energy-minimizing idle listening in wireless networks. In *Proceedings MobiCom '11*, 2011.

[26] X. Zhao, Y. Guo, Q. Feng, and X. Chen. A system context-aware approach for battery lifetime prediction in smart phones. In *Proceedings of SAC '11*, pages 641–646, New York, NY, USA, 2011. ACM.