M-026

WakeLock と Alarm の観察による無操作状態の消費電力の増加の原因となるアプリケーションの推定
Detecting Power Consuming Application based on WakeLock and Alarm Monitoring

栗原 駿† 福田 翔貴† 小柳 文乃‡
窪田 歩ξ 半井 明大ξ 小口 正人‡ 山口 実靖†
Shun Kurihara Shoki Fukuda Ayano Koyanagi
Ayumu Kubota Akihiro Nakarai Masato Oguchi Saneyasu Yamaguchi

1. はじめに

近年，スマートフォンやタブレット PC が普及し，それらの携帯端末で動作するソフトウェアプラットフォームとして Android OS が注目されている．また，スマートフォンの最大の課題は「バッテリの持続時間である」との報告があり[1]，Android OS における消費電力の低減は非常に重要な課題であると考えられる．

Android OS には指定時刻にアプリケーションを起動させる仕組みが用意されており，多くのアプリケーションにおいてユーザが直接操作を行わなくても動作する機能が備わっている．これらの機能による直接操作を行っていない状態でのアプリケーションの動作をユーザが把握することは困難であると予想できる．本稿では，Sleep 状態への移行を禁止する WakeLock 機能と，Sleep 状態でも起動する Alarm 機能に着目し，WakeLock の実行回数と Alarm の起動予定回数の観察による無操作時消費電力の大きいアプリケーションの推定手法を提案し，その評価結果を示す．

2. Android 端末における無操作時の電力消費

2.1 無操作状態における Android 端末の動作

無操作状態の Android 端末は Sleep 状態に入り，省電力モードとなる．Sleep 状態では，バッテリの主な消費原因である CPU 稼働，ディスプレイ点灯，通信が抑制される．

2.2 WakeLock

Sleep 状態への移行がアプリケーションの動作を妨げることを防ぐために，Android OS には Wake 状態端末が指定時間 Sleep しない(Wake 状態を保持する)ことを保証させる WakeLock という仕組みが用意されている．これは，センサで情報を取得し続ける，画面を点灯させ続けるなどの目的で使用される．WakeLock を多く行うアプリケーションは無操作時消費電力が大きいアプリケーションである可能性が高いと考えることができる．

2.3 AlarmManager

Android OS には，アプリケーションが将来のある時点に自身を起動させる仕組みである AlarmManager という機能がある．アプリケーションの起動をアラームとして OS に登録することにより，指定時刻における端末の状態(Sleep, Wake)に関わらず登録したアプリケーションを指定時刻に起動させることができる．起動時刻に端末が Sleep 状態である場合，必要に応じて端末が Wakeup され，アプリケーションが起動される[2]．この機能は，定期的に情報を取得するアプリケーションなどが指定時刻に更新や同期を行うためなどに利用される．Alarm を多く利用するアプリケーションも無操作時消費電力が大きいアプリケーションである可能性が高いと考えることができる．

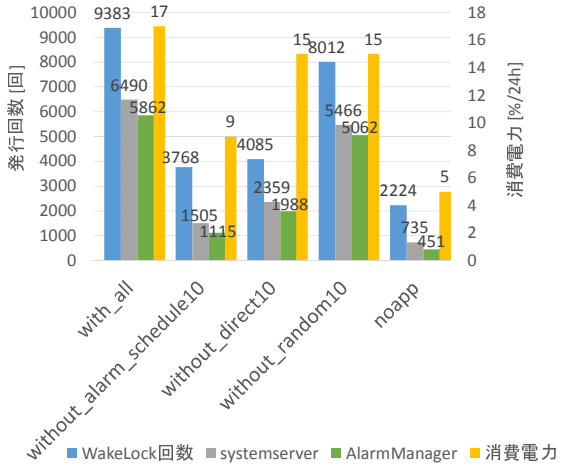

図 1 アプリケーション数と WakeLock と消費電力の関係

3. WakeLock と電力消費の関係

本章にて WakeLock と電力消費の関係について考察する．

3.1 測定方法

Android OS を改変し，無操作状態の Android 端末のバッテリ残量の推移と，アプリケーションが WakeLock の Release を行った時刻と，Release を行ったアプリケーションを調査した．測定は 24 時間(1440 分間)行い，測定対象アプリケーションセットは 2014 年 12 月 26 日における Google Play Store のウィジェット無料アプリケーションランキング[3]の上位 50 件とした．ただし，50 件の内 8 件のアプリケーションは端末に非対応であったため対象から外し，測定は 42 件のアプリケーションがインストールされている状態で行った．また，無操作状態となり 30 分後にディスプレイ表示がオフとなる設定とした．Android OS には，`onWakeLockAcquire()` メソッドと `onWakeLockReleased()`メソッドの呼び出しプロセスの PID とプロセス名，変数 `tag` の値，発行時刻を取得できるように改変が加えられている．具体的には `frameworks/base/services/java/com/android/server/power/Notifier.java` 内の `onWakeLockReleased()`メソッドにて上記情報を取得し，取得した情報を`/data/data/`内のテキストファイルに記録する修正を施している．

測定に用いた端末は Nexus7 (2013)，CPU Qualcomm Snapdragon S4 Pro 1.5GHz，メモリ 2GB，OS Android 5.0.1 である．

† 工学院大学
‡ お茶の水女子大学
ξ KDDI 研究所

3.2 測定結果

WakeLock 発行回数と消費電力の関係の調査結果を図 1 に示す．図の“WakeLock 回数”，“systemserver”，“AlarmManager”はそれぞれ，24 時間における端末全体の WakeLock 回数，そのうち systemserver プロセスが発行した WakeLock 回数，さらにその中でタグ名が AlarmManager であった WakeLock 回数を示している．これらが多ければ，より端末の Sleep が妨げられていることを意味している．“消費電力”は，24 時間におけるバッテリ残量の減少量[%]を示している．“with_all”は OS 標準(AOSP 配布 OS 添付の)アプリケーションに加え上記 42 件すべてのアプリケーションを端末に導入した状態を示し，“noapp”は端末に OS 標準のアプリケーションのみをインストールした状況を示す．“with_all”と“noapp”以外は次章にて述べる．測定結果より，全てのアプリケーションをインストールした環境では WakeLock が行われる回数が多く，無操作時消費電力も 17%と多いこと，アプリケーションを追加インストールしていない環境では WakeLock が行われる回数が少なく無操作時消費電力は 5%と少ないことがわかる．これより，無操作時消費電力の多くがインストールしたアプリケーションにより行われていることが分かり，無操作時消費電力の削減にはアプリケーションの動作の把握が重要であるとことが分かる．また，WakeLock 回数と無操作時消費電力には相関があると予想することができる．また，WakeLock 発行の多くは AlarmManager を用いた間接的な WakeLock であることがわかり，AlarmManager を用いた間接的な WakeLock 発行回数と無操作時消費電力に強い相関があると予想できる．

4. 無操作時消費電力の大きいアプリケーションの推定法の提案

4.1 推定手法

前章にて，WakeLock 発行回数や Alarm 回数と，無操作時消費電力に相関があるという予想を示した．本章では，Android OS を改変し WakeLock の発行と，AlarmManager によるアプリケーションの起動を調査し，これが多いアプリケーションを無操作時消費電力が大きいアプリケーションと推定する手法を提案する．

4.2 実装

Android OS に対し，`3.1` 節の改変と，アプリケーションが予定した `alarm` の起動予定時刻とそのアプリケーション名を記録するような改変を加えた．`alarm` の起動予定時刻とは `dumpsys alarm` コマンドによって得られる `alarm` の起動予定時刻である．具体的には，`frameworks/base/services/core/java/com/android/server/AlarmManagerService.java` 内の `dumpAlarmList()` メソッドで取得した上記情報を `/data/data/`内のテキストファイルに記録する修正を施した．

4.3 評価方法

前節の改変を施した `Android OS` を用いて，端末のバッテリ残量の推移と，アプリケーションが `WakeLock` を行った時刻と，アプリケーションが予定した `alarm` の起動予定時刻を調査，測定した．測定に用いた端末およびアプリケーションセットは `3.1` 節と同様である．

4.4 評価結果

WakeLock 発行回数と消費電力の関係は図 1 の通りである．“without_alarm_schedule10”は，alarm の起動予定回数の上位 10 件のアプリケーションをアンインストールした状態を示し，“without_direct10”は，アプリケーションの WakeLock 発行回数の上位 10 件のアプリケーションをアンインストールした状態を示す．“without_random10”は，全アプリケーションをアルファベット順に並べ 10 件アンインストールし測定を行う作業を 4 回行った結果の平均を示している．これらは，“with_all”より 10 件少ない 32 件のアプリケーションがインストールさている状態である．

測定結果より，“without_direct10”と“without_random10”を比べると WakeLock 回数は“without_direct10”の方が少ないが消費電力量は同等であることが分かる．“without_alarm_schedule10”と“without_direct10”および“without_random10”を比べると，“without_alarm_schedule10”が WakeLock 回数，消費電力量ともに大きく下回っていることがわかる．このことから， AlarmManager 使用の観察により無操作時消費電力の大きいアプリケーションを正しく推定できると評価することができる．

5. 考察

本手法の適用領域について考察を行う．本稿では改変した OS を用いて検証を行ったが， Alarm の観察は adb コマンド dumpsys alarm を用いている．よって非改変 OS，一般ユーザ権限でも本手法と同等の観察を行うことが可能であると考えられ，本手法を用いることで一般ユーザでも無操作状態の端末の電力消費に影響を及ぼすアプリケーションを把握することが可能であると考えられる．また WakeLock の観察には改変した OS が必要となるが，これらはアプリケーション開発者やアプリケーションマーケットの運営者がリファレンス端末などを用いてアプリケーション評価環境を構築し，自身が開発したアプリケーションや自サイトで配布するアプリケーションの評価を行う使用法なども有効な適用領域であると期待できる．

6. おわりに

本稿では，無操作状態の Android 端末における消費電力に着目し，WakeLock の実行回数や Alarm のセット回数に基づく無操作時消費電力の多いアプリケーションの推定手法を提案した．また，本手法により推定されたアプリケーションをアンインストールした結果大幅な消費電力の低減が確認され，本手法の有効性が確認された．

参考文献

[1] 日本経済新聞 2013 年 4 月 1 日
http://www.nikkei.com/article/DGXNASFK2600W_W3A320C1000000/
[2] AlarmManager
http://developer.android.com/reference/android/app/AlarmManager.html#
[3] GooglePlayStore 無料アプリケーションランキング
https://play.google.com/store/apps/category/APP_WIDGETS/collection/topselling_free