M-005

複数機能のソフトウェア的依存性を考慮したアプリケーションごとの消費電力の推定

Application Power Consumption Estimation Considering Software Dependency Supporting Multiple Functions in Android

栗原 駿† 福田 翔貴† 小口 正人‡ 山口 実靖†
Shun Kurihara Shoki Fukuda 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，これらは重要な情報端末プラットフォームとなっている．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[1]があり，アプリケーション毎の消費電力の把握は，ユーザやアプリケーションマーケット運営者にとって重要である．また，Android OS では無操作状態でもアプリケーションが動作し電力を消費する．無操作状態におけるアプリケーションの動作の把握は特に困難であり，その電力消費の把握は重要であると考えられる．

しかし，アプリケーションのインストールやアンインストールによる消費電力の増減の量は端末に依存し，あるアプリケーション消費電力の大小を一概に決めることはできない．端末依存性としては，ハードウェア的依存性(端末のハードウェア構成による影響)とソフトウェア的依存性(端末にインストールされているアプリケーション構成の影響)があり，消費電力の見積もりにはこれらを考慮することが必要となると考えられる．我々は過去に，GPS を使用するアプリケーション群を対象にソフトウェア依存性を考慮した GPS 時間推定手法[2][3]を提案し，その GPS 時間に基づき消費電力を推定し評価を行った．この手法は，アプリケーションの電力消費の主たる要因が GPS の場合には有効であるが，それ以外の要因(WakeLock，CPU，Wi-Fi など)の場合は有効性が確認できていない．

本稿では，端末のソフトウェア的依存性を考慮したアプリケーション毎の消費電力見積もり手法の一つとして，WakeLock による消費電力の増加の見積もり手法を提案し，GPS のみでなく複数の機能を考慮したソフトウェア的依存性を考慮した見積もりの実現を目指す．そして，評価によりその有効性を示す．

2. 既存研究

Broadcast Intent に着目し，端末のソフトウェア的依存性について考察した研究として文献[4]や文献[5]がある．アプリケーションの起動方法の 1 つに Broadcast Intent があり，各アプリケーションの動作回数は発行される Broadcast Intent 回数に依存する．これらの研究では，Broadcast Intent の発行回数や 1 発行あたりのアプリケーションの動作や消費電力の調査を行っている．

3. Android における無操作時電力消費と見積もり

3.1 Android 端末における無操作時の電力消費

無操作状態における 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．また，Android OS には，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の WakeLock 機能が用意されている．WakeLock は，端末が指定時間 Sleep しない(Wake 状態を保持する)ことを保証させる仕組みである．これは主に，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．

† 工学院大学 電気・電子工学専攻
‡ お茶の水女子大学 理学部 情報科学科

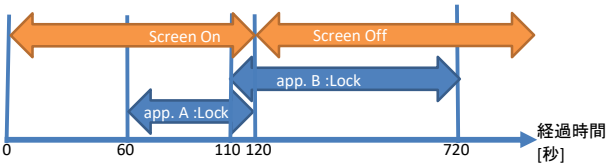

図 1 測定対象アプリケーションの動作

この機能により，無操作状態であっても Android 端末は必ずしも Sleep 状態に入れるとは限らない．そのため，無操作状態におけるアプリケーション毎の消費電力を算出するには，正確にアプリケーション毎の WakeLock 時間を把握する必要がある．

3.2 Android OS における消費電力の見積もり

Android OS 標準のアプリケーション毎の消費電力量集計機能(設定アプリケーションの電池機能，以降 Android 手法と呼ぶ)では，アプリケーション毎の CPU 時間や GPS 時間，WakeLock 時間の累積により各アプリケーションの消費電力を求めている．

4. 基礎性能調査

本章にて，Android 手法で用いられるアプリケーション毎の WakeLock 時間の精度を調査する．

4.1 調査方法

Android 手法により算出された端末全体の WakeLock 時間とアプリケーション毎の WakeLock 時間を調査した．WakeLock 時間は Android OS 標準の設定アプリケーションの電池機能の実装内部にて管理されている．同実装を改変しアプリケーション外部から取得できるようにした．そして，その取得結果を，あるアプリケーションのアンインストールにより減少する WakeLock 時間と考え，端末全体の WakeLock 時間を推定した．また，実際にそのアプリケーションをアンインストールしたあとの端末全体の WakeLock 時間を実測し，両値を比較した．測定は 30 分間とし無操作状態から 2 分後にディスプレイが OFF になる設定とした．

測定は自作のアプリケーションを 2 件動作させて行った．測定対象アプリケーション 2 件は，ほぼ同時に起動され，

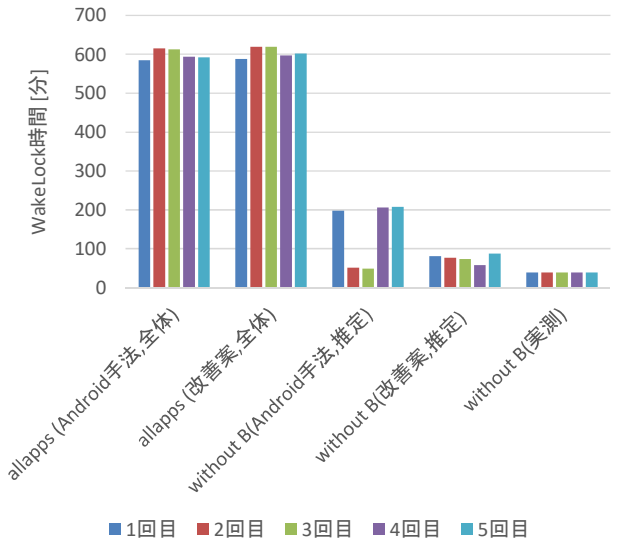

図 2　測定 5 回の WakeLock 時間

ほぼ同時にバックグラウンド状態に移行する．移行した時点を測定開始とした．バックグラウンドに移動後の対象アプリケーションの動作は，図 1 の通りである．すなわち，App. A は，60 秒後に 60 秒間の WakeLock を行う．App. B は，110 秒後に 10 分間の WakeLock を行う．調査用の Android OS は，calculateApp() メソッドと calculateRemaining()メソッドにて，UID と発行時刻と端末全体の WakeLock 時間とアプリケーション毎の WakeLock 時間を取得できるように改変を加えた．測定に用いた端末は Nexus7 (2013)，CPU は Qualcomm Snapdragon S4 Pro 1.5GHz，メモリサイズは 2GB，OS は Android 6.0.1 である．

4.2 調査結果

測定結果を図 2 に示す．“allapps(Android 手法,全体)”は，全アプリケーションをインストールした状態で Android 手法が報告した端末全体の WakeLock 時間を示し，“without B(Android 手法,推定)”は allapps(Android 手法,全体)からアプリケーション B をアンインストールした時の端末全体の WakeLock 時間を Android 手法の報告値から算出した推定値を示し，“without B(実測)” は，実際にアプリケーション B をアンインストールした状態で測定し報告された端末全体の WakeLock 時間を示す．他は，次章にて用いる．

図 2 より，without B(Android 手法，推定)と without B(実測)を比較すると，2 回目，3 回目の測定では，実測値と非常に近い値を推定できているが，1 回目，4 回目，5 回目の測定では推定値と実測値が大きく異なっていることが分かる．このことから，現実装の Android 手法のアプリケーション毎の WakeLock の観察は，複数アプリケーションにより WakeLock が重なった場合にその重なった WakeLock 時間を案分し各アプリケーションに配分するなどの処理を含んでおり[3]，これらにより現実装は必ずしも正確な見積もりになっていないと考えられる．

5. アプリケーション毎の WakeLock 時間算出方法の改善案

本章では，アプリケーション毎の WakeLock 時間算出方法の改善案の提案を行う．

改善案では，Android OS の実装に対して WakeLock の *acquire* と *remove* の観察が可能となるように修正を行う．これは，`BatteryStatsImpl.java` の修正により可能である．そして，アプリケーションを同 OS 上で動作させ，アプリケーション毎の *acquire* 発行時刻と *remove* 発行時刻を記録する．最後に，この記録に基づき，端末全体の WakeLock 時間，アプリケーション毎の WakeLock 時間，複数アプリケーションによる WakeLock が重なっている時間を取得し，各アプリケーションのアンインストール時の WakeLock の減少量を推定する．記録から推定する方法は既存手法[3]と同様であり，WakeLock 記録から特定のアプリケーションを削除し，それを推定結果とする．

6. 性能評価

4.1 節と同様の手法で推定精度を評価した．測定結果を，図 2 に示す．図 2 の “allapps(改善案，全体)”は，全アプリケーションをインストールした状態で改善案により算出した端末全体の WakeLock 時間を示し，“without B(改善案，推定)”は，allapps(改善案,全体)からアプリケーション B をアンインストールした時の端末全体の WakeLock 時間を改善案から算出した推定値を示す．

without B(Android 手法,推定)と without B(改善案,推定)を比較すると，1 回目，4 回目，5 回目は改善案の精度の方が大幅に上回り，2 回目，3 回目では Android 手法の精度の方がわずかに高い結果となった．

7. おわりに

本稿では，Android OS 標準のアプリケーション毎の WakeLock 時間算出方法に対しての改善案を提案し，評価により有効性を確認した．今後の予定は，実アプリケーションを用いた改善案の評価と Android OS 標準のアプリケーションの実装の改変をしていく予定である．

謝辞

本研究は JSPS 科研費 26730040, 15H02696, 17K00109 の助成を受けたものである．

本研究は、JST、CREST JPMJCR1503 の支援を受けたものである．

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] Shun Kurihara, Shoki Fukuda, Shintaro Hamanaka, Masato Oguchi, Saneyasu Yamaguchi, “Application Power Consumption Estimation Considering Software Dependency in Android ”, ACM IMCOM 2017, 2017/1

[3] 栗原 駿, 福田 翔貴, 濱中 真太郎, 山口 実靖, 小口 正人, “ソフトウェア依存性を考慮した断続的起動アプリケーションの消費電力推定”, IPSJ79,2017/3

[4] 早川 愛, 半井 明大, 竹森 敬祐, 山口 実靖, 小口 正人, “Android 端末省電力化に向けたブロードキャストインテント発行とアプリケーションの因果関係の評価” , インターネットコンファレンス(IC2014), 2014/11

[5] 中村 優太, 早川 愛, 半井 明大, 竹森 敬祐, 小口 正人, 山口 実靖, “Android 端末におけるインストールアプリケーションとブロードキャストインテント発行による電力消費に関する一考察”, DBS モバイル・地理情報システム, 2014/8