Vol.2018-GN-103 No.7
Vol.2018-CDS-21 No.7
Vol.2018-DCC-18 No.7
2018/1/26

Android OS における アプリケーション毎の消費電力推定手法の実アプリケーション評価

栗原駿[†1]　福田翔貴[†1]　神山剛[†2]　福田晃[†2]　小口正人[†3]　山口実靖[†1]

概要：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，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,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的依存性を考慮した GPS 使用時間推定手法を提案し，その推定 GPS 使用時間に基づく消費電力の推定を行った．しかし，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，それ以外の要素(WakeLock，CPU，Wi-Fi など)が主たる要因の場合は有効でない．本稿では，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，アプリケーション毎の WakeLock 時間の見積もりとそれに基づいた消費電力推定手法を提案し，GPS のみでなく複数の機能の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，使用モデルに基づく実アプリケーションセットを用いて性能評価を行い，提案手法の有効性を示す．

キーワード：Android，WakeLock

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．ユーザが関与していない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)が主たる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，使用モデルに基づく実アプリケーションセットを用いて性能評価を行い，その有効性を示す．

†1 工学院大学
†2 九州大学
†3 お茶の水女子大学

2. 関連研究

Android OS における消費電力推定に関する研究としては，以下のものがある．Corral らは，Android OS から得られるパラメータに基づいて消費電力を推定する手法について考察している[4]．Kaneda らは，無線通信機器の消費電力を解析するための消費電力モデルとその生成方法について述べている[5]．これらの論文では，各アプリケーションの消費電力推定方法については考察されていない．文献[6]は，WakeLock 発行回数に基づいて消費電力推定手法を提案している．しかし，ソフトウェア依存性は考慮されていない．

Android OS は，GPS，WakeLock，Wi-Fi，CPU などの電力消費源の使用時間を合計することによって，各アプリケーションの消費電力を算出している．そのため，各アプリケーションの消費電力の推定精度を向上するには，電力消費源の使用時間の精度を向上させることが重要である．各アプリケーションの GPS 利用時間の精度を向上させる方法が，文献[2]にて提案されている．

消費電力のソフトウェア的依存性に関する研究としては，以下のものがある．文献[7]，[8]では，Broadcast Intent 発行による通信量の増加や，それによる消費電力増加に関する考察が行われている．文献[9][10]では，既存手法[2]を WakeLock 機能に適用しベンチマークアプリケーションを用いて評価が行われている．しかし，実アプリケーション

Vol.2018-GN-103 No.7
Vol.2018-CDS-21 No.7
Vol.2018-DCC-18 No.7
2018/1/26

を用いた評価はされていない．

文献[11]では，一般性のあるスマートフォン利用モデルを提案している．この研究では，スマートフォンユーザに対するアンケート調査結果や端末ログ収集調査の情報などによるデータを用いることで，実際のスマートフォンユーザのスマートフォン利用を，ユーザの属性やアプリケーション使用傾向などの特徴とともに示している．

3. Android OS における無操作状態電力消費とその見積もり

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．

この機能により，無操作状態であっても Android 端末は Sleep 状態に入らないことがある．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，アプリケーション毎の WakeLock 時間を正確に把握することが重量である．WakeLock は，*acquire* でロックし *release* でロックを解放する．

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，GPS 時間，WakeLock 時間などを求め，それらの累積により各アプリケーションの消費電力を計算している．

3.3 IMCOM2017 手法による GPS 時間の見積もり

本節では，IMCOM2017 手法について説明する．具体的には，GPS 使用履歴からあるアプリケーションのアンインストール後の端末全体の GPS 使用時間の推定について説明する．図 1 のように，アプリケーションが GPS を使用し，この状態から “App. B”のアンインストール後の端末全体の GPS 使用時間を見積もる．

まず，“App. A”の GPS 使用時間は 8 となる．“App. B”の GPS 使用時間も 8 となる．次に端末全体の GPS 使用時間は，App. A の GPS 使用時間と App. B の GPS 使用時間の論理和となるので，10 となる．最後に，App. B のアンインストール後の GPS 使用時間は，App. B の GPS 使用履歴を削除した状態の端末全体の GPS 使用時間となるので 8 となる．

表 1　他のアプリケーションアンインストール前後における GMS の WakeLock 時間及び発行回数

	アンインストール前		アンインストール後	
	時間 [秒]	回数	時間 [秒]	回数
パターン 1	238.1	3450	223.3	3252
パターン 2	480.8	3925	289.7	3866
パターン 3	259.5	3515	236.7	3139
パターン 4	240.7	3274	214.2	3055
パターン 5	377.4	4986	260.8	3646
パターン 6	241.9	3356	284.2	3846

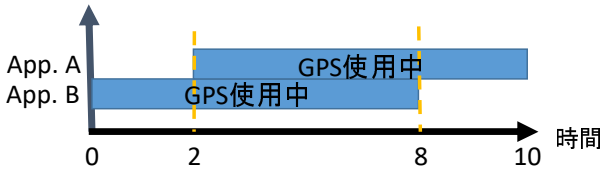

図 1 アプリケーション毎の GPS 使用時間

4. システムプロセスによる WakeLock 発行のソフトウェア依存性

WakeLock は，ユーザがインストールしたアプリケーション以外に，システムプロセスによっても発行される．特に Google 開発者サービス(以降 GMS と呼ぶ)プロセスによっては，多くの WakeLock が発行される．GMS プロセスによる WakeLock の発行は，OS としては GMS プロセスからの発行として認識されるが，GMS プロセスがアプリケーションからの要請がなく独自に発行することは少なく，実際は何らかのアプリケーションからのサービス使用要求の結果として WakeLock が発行されていることが多いと考えられる．よってあるアプリケーションをアンインストールすると，当該アプリケーションによる WakeLock 発行が消滅するとともに，当該アプリケーションからの要請を受けて GMS が発行していた WakeLock も消滅することになる．すなわち，GMS による WakeLock 発行にソフトウェア的依存性があり，あるアプリケーションをアンインストールしたときに減少する端末全体の WakeLock の量を正確に推定するには，GMS による発行のソフトウェア的依存性を考慮して推定する必要があると考えられる．

表 1 は，6 種類のアプリケーションセットにおいて，その中からあるアプリケーションをアンインストールした場合におけるアンインストールの前と後の GMS プロセスによる 24 時間での WakeLock の回数と時間を表している．アプリケーションセットのパターン 1 から 6 は，性能評価の章にて説明をする．表より，あるアプリケーションをアンインストールすると GMS による WakeLock の回数と時間が変化していることが分かる．これらの例も，WakeLock 時間の推定には GMS の WakeLock 発行のソフトウェア的依存性を考慮することが重要であるとの仮説を支持していると言

Vol.2018-GN-103 No.7
Vol.2018-CDS-21 No.7
Vol.2018-DCC-18 No.7
2018/1/26

表 2　測定対象アプリケーション群 (1/2)

パターン 1	パターン 2	パターン 3
ライフスタイル 1	メール	ゲーム 1
写真 1	SNS1	SNS2
	SNS2	写真 2
	セキュリティ	健康 2
	ブラウザ	セキュリティ

表 3　測定対象アプリケーション群 (2/2)

パターン 4	パターン 5	パターン 6
SNS2	メール	ライフスタイル
SNS3	SNS2	ライフスタイル 3
メール	ゲーム 2	ツール 3
ツール	ゲーム 3	ツール 4
健康 2	ゲーム 4	ブラウザ
	通信	セキュリティ
	ライフスタイル 2	
	ツール	
	ツール 2	
	メディア	

える．
ただし，パターン 1 から 5 はアプリケーションをアンインストールすることにより GMS による発行が減少する直感に近い例となっているが，パターン 6 はアンインストールにより GMS による発行が増加する直感に反する特異な例となっている．

5. WakeLock 履歴に基づいた推定手法の提案

本章では，アプリケーション毎の WakeLock を観察できる様に Android OS を改変し，その WakeLock 観察履歴に基づき，あるアプリケーションのアンインストール後の WakeLock 時間を推定する手法を 4 つ提案する．

本手法では，Android OS の実装に対してアプリケーション毎の WakeLock の acquire と release 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 WakeLock の acquire 発行時刻と release 発行時刻を記録する．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．

1 つ目の IMCOM2017 手法は，既存手法[2]の手法をナイーブに適用し，WakeLock が重なっている時間を案分することによるずれを考慮した手法である．ただし，対象アプリケーションをアンインストールすることにより当該アプリケーション以外から(例えば GMS などのシステムプロセスから)発行される WakeLock の数が変化することは考慮していない．2 つ目の連鎖手法は，対象アプリケーションをアンインストールすることにより GMS により発行される WakeLock 数が減少する，そして当該アプリケーションの WakeLock 中に発行された GMS プロセスの WakeLock は当該プロセスに起因する(当該アプリケーションのアンインストールにより消滅する)という仮説に基づいている．3 つ

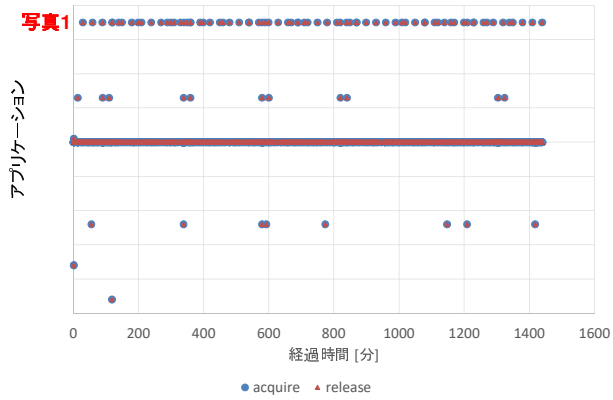

図 2 パターン 1 の WakeLock 履歴

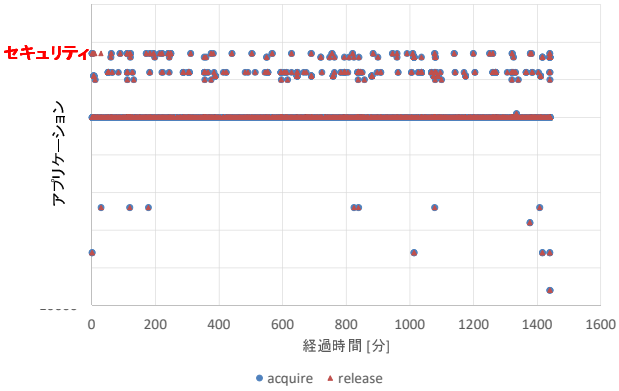

図 3 パターン 2 の WakeLock 履歴

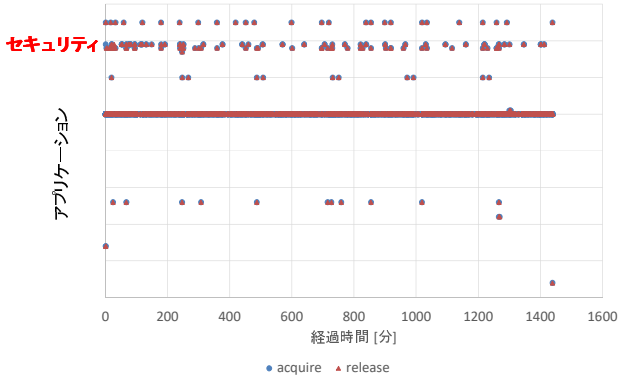

図 4 パターン 3 の WakeLock 履歴

Vol.2018-GN-103 No.7
Vol.2018-CDS-21 No.7
Vol.2018-DCC-18 No.7
2018/1/26

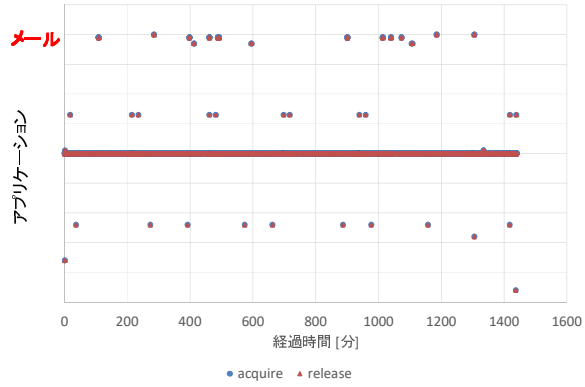

図 5　パターン 4 の WakeLock 履歴

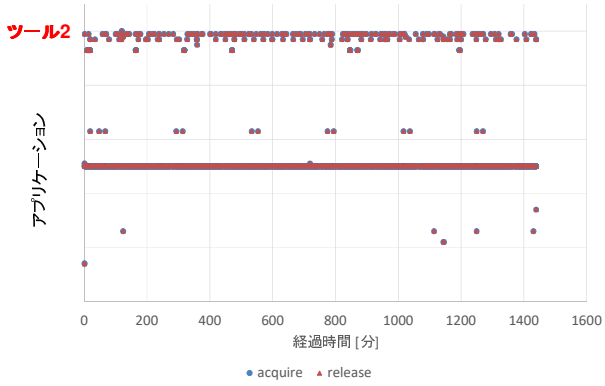

図 6　パターン 5 の WakeLock 履歴

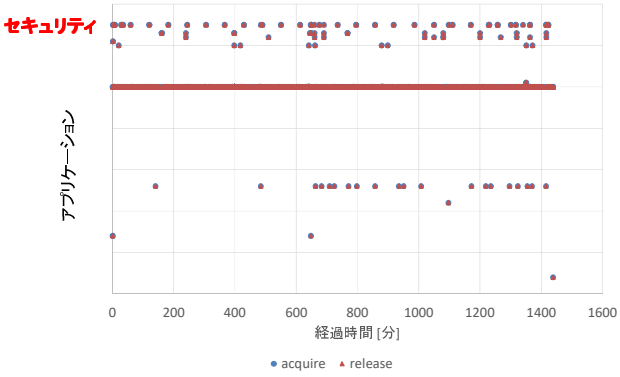

図 7　パターン 6 の WakeLock 履歴

目と 4 つ目は，アンインストールにより GMS による WakeLock が減少すると仮定し，その量をインストールアプリケーション毎の WakeLock 発行の時間と回数の比例配分により推定している手法である．

5.1 IMCOM2017 手法

本手法は，アプリケーション毎の WakeLock 履歴を基に既存研究[2]と同様の方法で推定を行う．具体的には，端末全体の WakeLock 履歴からアンインストール対象アプリケーションの WakeLock 履歴を削除し，端末全体の WakeLock 時間を推定する．本手法は GMS の WakeLock の変化は考慮していない．

5.2 連鎖手法

本手法は，IMCOM2017 手法に加えて，測定対象アプリケーションの WakeLock 中に開始(acquire の発行)がされた GMS の WakeLock も削除し，推定を行う．

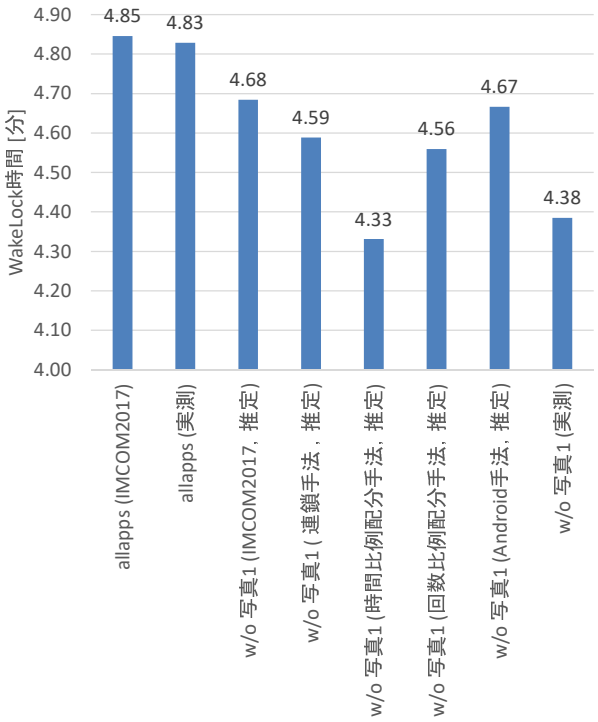

図 8　パターン 1 の推定結果

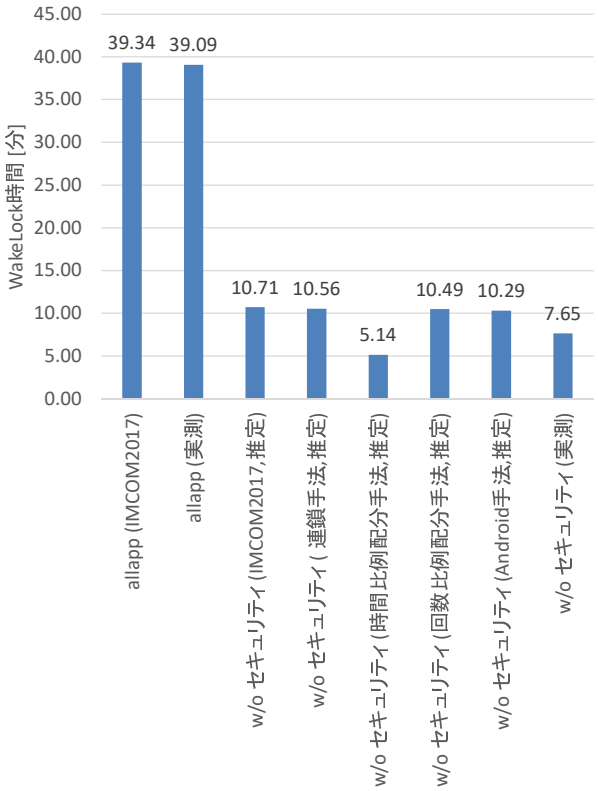

図 9　パターン 2 の推定結果

5.3 時間比例配分手法

本手法は，IMCOM2017 の手法に加えて，以下の手順に従い GMS による WakeLock を破棄率 p で削除し，推定を行う．

Vol.2018-GN-103 No.7
Vol.2018-CDS-21 No.7
Vol.2018-DCC-18 No.7
2018/1/26

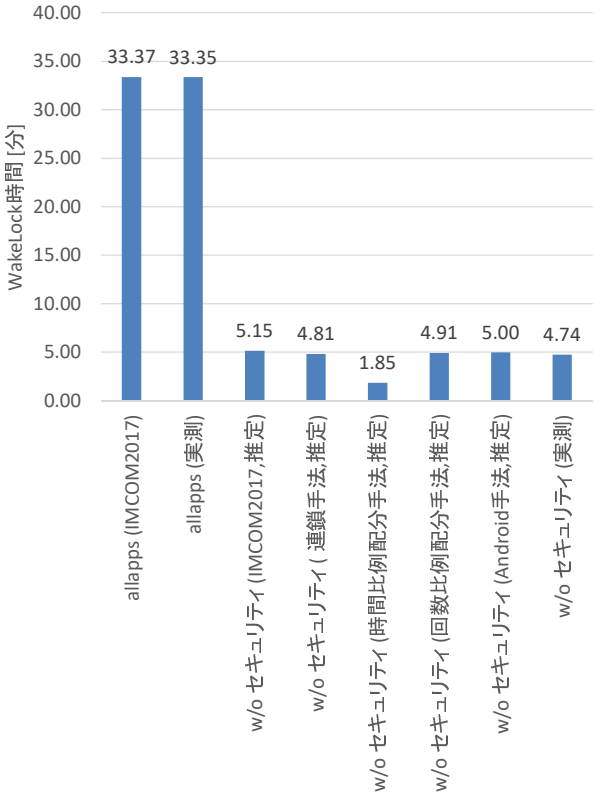

図 10　　パターン 3 の推定結果

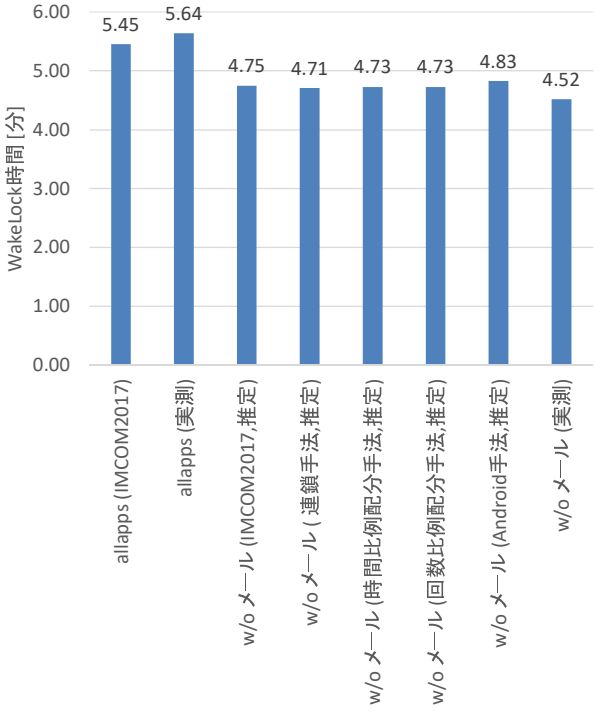

図 11　　パターン 4 の推定結果

アンインストール対象アプリケーションの WakeLock 時間, GMS と対象アプリケーション以外のアプリケーションの WakeLock 時間の合計の割合 p を算出する．回数比例配分手法

本手法は，IMCOM2017 の手法に加えて，以下の手順に従い GMS による WakeLock を破棄率 p で削除し，推定を行う．

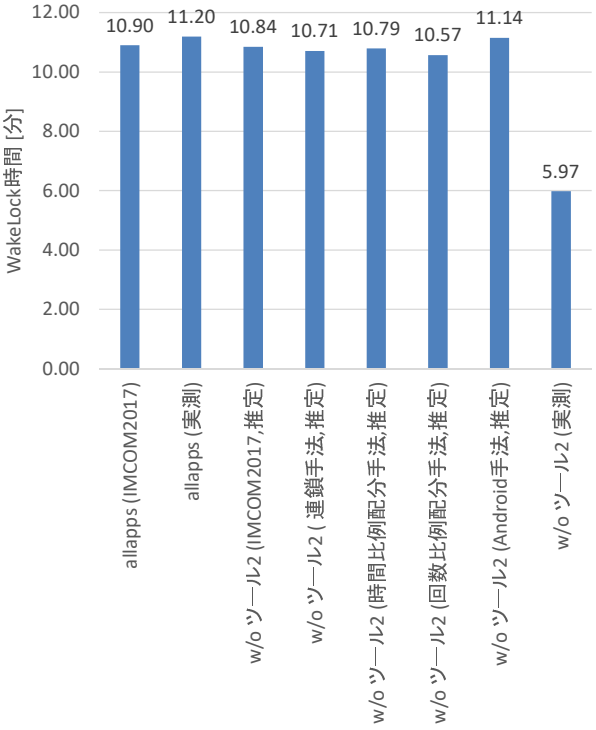

図 12　　パターン 5 の推定結果

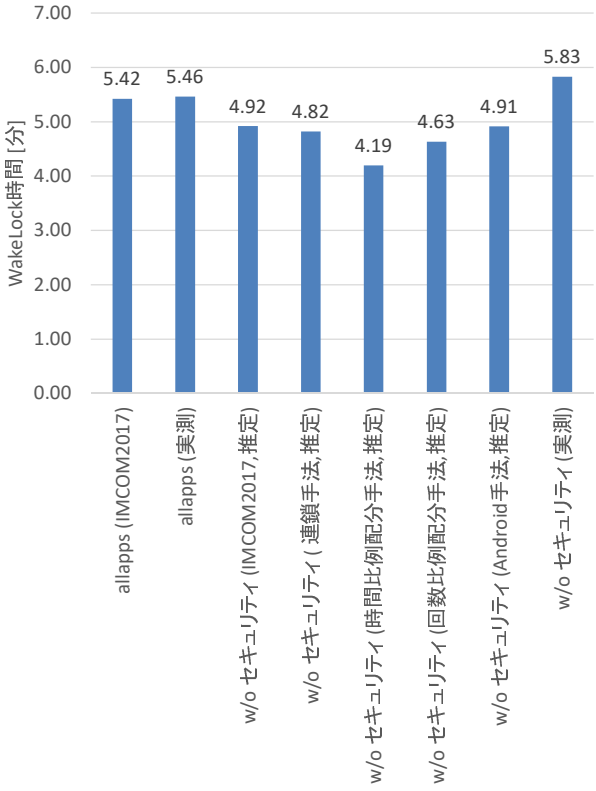

図 13　　パターン 6 の推定結果

アンインストール対象アプリケーションの WakeLock 回数, GMS と対象アプリケーション以外のアプリケーションの WakeLock 回数の合計の割合 p を算出する．

Vol.2018-GN-103 No.7
Vol.2018-CDS-21 No.7
Vol.2018-DCC-18 No.7
2018/1/26

6. 性能評価

本章では，提案手法と Android 手法の推定精度を比較し性能評価を行う．測定方法は，測定対象アプリケーション群をそれぞれの端末にインストールを行い，24 時間無操作状態で放置して WakeLock の発行を観察する．各手法によりアンインストール対象アプリケーションのアンインストール後の WakeLock 時間の減少量を推定して，実際の WakeLock 時間減少量と比較して推定の正確度の評価を行う．アンインストール対象アプリケーションは，Android 手法によって消費電力が最も多いと報告された測定対象アプリケーションとする．測定対象アプリケーションセットは，表 2，3 の通りである．これらは，文献[11]をもとに作成をした．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．

今回の測定における，各パターンのアンインストール対象アプリケーションは，パターン 1 が写真 1，パターン 2 がセキュリティ，パターン 3 がセキュリティ，パターン 4 がメール，パターン 5 がツール 2，パターン 6 がセキュリティとなった．図 2〜7 に，各パターンのアプリケーション毎の WakeLock 履歴を示す．次節以降にて各手法の推定精度について述べる．

6.1 IMCOM2017 手法の評価結果

本節では，IMCOM2017 手法の性能評価を行う．
推定結果を，図 8 から 13 のラベル(IMCOM2017 手法，推定)に示す．図の “allapp (IMCOM2017)”は,各パターンにおいて測定対象アプリケーションがすべてインストールされた状態で IMCOM2017 手法により算出された端末全体での WakeLock 時間である． “allapps (実測)”は，各パターンにおいて測定対象アプリケーションがすべてインストールされた状態で Android 手法により算出された端末全体での WakeLock 時間である． “w/o (IMCOM2017 手法，推定)”は，IMCOM2017 手法によりアプリケーションアンインストール後の端末全体の WakeLock 時間を推定した推定値である．また，“w/o (Android 手法，推定)”は，Android 手法によりアプリケーションアンインストール後の端末全体の WakeLock 時間を推定した推定値である． “w/o (実測)”は，実際に対象アプリケーションをアンインストールした時の実測値である．この値は正解値であり，この値と近い推定値が良い推定値となる．他のラベルのデータは次節以降にて説明する．

IMCOM2017 手法と Android 手法を比較するとわずかに IMCOM2017 手法の推定精度の方が悪い結果となった．また，パターン 6 では，アンインストール後に WakeLock 時間が長くなるという直感と異なる結果になり，両手法ともに推定が大きく外れている．

6.2 連鎖手法の評価結果

本節では，連鎖手法の性能評価を行う．図 8〜13 の “w/o (連鎖手法，推定)”は，連鎖手法によりアプリケーションアンインストール後の端末全体での WakeLock 時間を推定した推定値である．

各パターンにおいて連鎖手法と Android 手法を比較するとパターン 1，3，4，5 において，Android 手法よりわずかに高い正確度で推定できることを確認した．

6.3 時間比例配分手法の評価結果

本節では，時間比例配分手法の性能評価を行う．図 8〜13 の “w/o (時間比例配分手法，推定)”は，時間比例配分手法によりアプリケーションアンインストール後の端末全体での WakeLock 時間を推定した推定値である．

各パターンにおいて時間比例配分手法と他の手法と比較すると，パターン 1 では実測値と同程度の値を推定できているが，パターン 3 では他の手法に比べて正確度が低くなっていることが分かる．

6.4 回数比例配分手法の評価結果

本節では，回数比例配分手法の性能評価を行う．図 8〜13 の “w/o (回数比例配分手法，推定)”は，回数比例配分手法により，アプリケーションアンインストール後の端末全体での WakeLock 時間を推定した推定値である．

各パターンにおいて，回数比例配分手法と Android 手法を比較すると，パターン 1，3，4，5 では，わずかに高い正確度で推定できていることが分かる．

7. 考察

前章の性能評価により，WakeLock の重なりのみを考慮した手法(IMCOM2017 の手法)においては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 量を正確に推定することができず，実際の WakeLock 量は推定値よりもさらに少ない(減少量が推定より大きい)状態になることが分かった．そして，前述の表 1 の結果が示すように，GMS の発行数がアプリケーションのアンインストールにより減少しており，これを考慮して推定した手法(連鎖手法，時間比例配分手法，回数比例配分手法)の法がより正確度の高い推定が行えていることが分かる．

本稿では，対象アプリケーションが WakeLock 中に開始された GMS による WakeLock は対象アプリケーションに起因する WakeLock であるという仮説や，GMS 以外の WakeLock 発行の時間や回数に比例するという仮説を立て，それに基づく推定手法を提案している．これは，以下に述べるような方法に依りより正確度が高い推定に改善できると考えられる．すなわち，オペレーティングシステムを改変し，アプリケーションプロセスから GMS プロセスへの通信(サービス使用の要求)をモニタリングし，GMS プロセスによる WakeLock 発行の原因となるプロセスを明確にする．そして，あるアプリケーションのアンインストール時

Vol.2018-GN-103 No.7
Vol.2018-CDS-21 No.7
Vol.2018-DCC-18 No.7
2018/1/26

にはそのアプリケーションに起因する GMS プロセスの WakeLock も消滅すると推定する.

8. おわりに

本稿では，消費電力の端末依存性，特にアプリケーションのアンインストールによる GMS の WakeLock の減少を考慮したアプリケーション毎の WakeLock 時間推定手法を提案した．そして，実使用モデルに基づくアプリケーションセットに用いて推定の正確度を評価した．評価の結果，提案手法は実モデルアプリケーションセットにおいても有効であることが確認された． 今後は GMS による WakeLock の変化のより正確な推定手法についての考察を行っていく予定である.

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] S. Doki, T. Ogishi, S. Ano, "Mobile interface control scheme can extend battery life," *2015 International Conference on Information Networking (ICOIN)*, Cambodia, 2015, pp. 116-121. doi: 10.1109/ICOIN.2015.7057867

[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] Shun Kurihara, Shoki Fukuda, Masato Oguchi, Saneyasu Yamaguchi, "Estimation of Power Consumption of Each Application Based on Software Dependency in Android", GCCE2017, 2017/10

[4] Luis Corral, Anton B. Georgiev, Alberto Sillitti, and Giancarlo Succi, "A method for characterizing energy consumption in Android smartphones," Green and Sustainable Software (GREENS), 2013 2nd International Workshop on, San Francisco, CA, USA, 20-20 May 2013.

[5] Y. Kaneda, T. Okuhira, T. Ishihara, K. Hisazumi, T. Kamiyama, M. Katagiri, "A run-time power analysis method using OS-observable parameters for mobile terminals," Proc. ICESIT, pp. 1-6, 2010.2010 (1) , p.39 , 2010-02

[6]S. Kurihara, S. Fukuda, A. Koyanagi, A. Kubota, A. Nakarai, M. Oguchi, and S. Yamaguchi, "A Study on Identifying Battery-Draining Android Applications in Screen-Off State," 2015 IEEE 4th Global Conference on Consumer Electronics, 2015-10.

[7] 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

[8] 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

[9] 栗原 駿, 福田 翔貴, 小口 正人, 山口 実靖, "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", FIT2017, 2017/9

[10] Shun Kurihara, Shoki Fukuda, Masato Oguchi, Saneyasu Yamaguchi, "Estimation of Power Consumption of Each Application Caused by Device Lock Considering Software Dependency in Smartphones", CANDAR'17, 2017-10

[11] Takeshi Kamiyama, Kenji Hisazumi, Hiroshi Inamura, Teppei Konishi, Ken Ohta, Akira Fukuda, "Smartphone Usage Analysis Based on Actual-Use Survey", MobiCASE'16 Proceedings of the 8th EAI International Conference on Mobile Computing, Applications and Services, Cambridge, Great Britain — November 30 - December 01, 2016