Vol.2015-GN-93 No.22
Vol.2015-CDS-12 No.22
Vol.2015-DCC-9 No.22
2015/1/27

Android OS における電力消費を誘発するアプリケーションに関する一考察

中村優太[†1]　早川愛[†2]　小柳文乃[†2]　半井明大[†3]
竹森敬祐[†3]　小口正人[†2]　山口実靖[†1]

スマートフォンユーザの約 7 割がバッテリの持続時間に不満を持っているとの報告があり，主要なスマートフォン OS である Android を搭載した端末の消費電力の低減は重要な課題と言える．Android 搭載端末では，無操作状態の端末でアプリケーションが起動され，端末の電力が消費されることがある．しかし，ユーザの直接的操作に起因しない無操作状態端末でのアプリケーションの動作や電力消費を把握することは困難である．本稿では，Android OS を改変して無操作状態において端末やアプリケーションの起動を誘発するアプリケーションの観察を行い，無操作状態端末の電力消費を誘発するアプリケーションについての考察を行う．

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．Android OS の世界市場における 2014 年第 3 四半期のシェアは 84.4%に達しており[1]，Android OS は非常に重要なプラットフォームとなっている．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[2]，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 は PC 用 OS である Linux を元に作成されており，Linux OS に類似した動作をする．すなわち，Android OS はマルチタスク OS であり常に複数のプロセスが常駐，動作している．また，指定時刻やイベント発生時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．これらのユーザの直接的操作を伴わないアプリケーションの起動により端末のバッテリが消費されるが，ユーザが直接操作をしていない状態における端末やアプリケーションの動作の把握はユーザには困難であると予想できる．

2. Android 端末における電力消費

Android 端末の電力消費はユーザが操作しているときに消費される電力と，ユーザが操作をしていないときに消費される電力に分類することができる．無操作時にアプリケーションを起動する仕組みのひとつにブロードキャストインテントがある．ブロードキャストインテントについては 2.1 節で紹介する．無操作時が続くと Android 端末は Sleep 状態に入り，省電力モードに入る．端末を Sleep 状態にさせることなく指定の処理の実行を保証する仕組みとして WakeLock がある．WakeLock については 2.2 節で紹介する．

2.1 ブロードキャストインテント

Android にはインテントと呼ばれる仕組みが用意されており，アプリケーション同士の連携やシステム-アプリケーション間の連携はインテントを用いて行われる．

インテントには明示的インテント，暗黙的インテント，ブロードキャストインテントがある．明示的インテントと暗黙的インテントは，ユーザがアプリケーション内のボタンをタップしたときなどに発行され，主にフォアグラウンドアプリケーションを別のアプリケーションに切り替えるときに使用される．

ブロードキャストインテントは Android システムやアプリケーションから全てのアプリケーションに対して放送型で発行され，各アプリケーションは自身が受信するブロードキャストインテントと当該インテント受信時に実行する処理を定義することができる．ブロードキャストインテントを受信登録しているアプリケーションはレシーバと呼ばれ，本稿では受信の登録をレシーバ登録と呼ぶ．レシーバには予め Android Manifest に記載しておく静的な登録方法とアプリケーションが必要な時に登録する動的な登録方法がある．

Android OS には公式に定められているイベントと当該イベント発生時に発行されるブロードキャストインテントが定義されている．例えば OS 起動時には BOOT_COMPLETED インテントが，バッテリ残量変化時には BATTERY_CHANGED インテントがシステム(system_server プロセス)により発行される．アプリケーション開発者がこれらイベントの発生時に行う処理を定義したい場合は，対応するインテントをレシーバ登録し，受信時の処理を実装する．

Android OS では，ユーザが直接的に端末を操作していない場合でもブロードキャストインテントが発行されており(発行頻度は 3 章にて述べる)，これを契機に CPU 処理や通信などが行われ，結果として電力が消費されている．

†1 工学院大学大学
Kogakuin University
†2 お茶の水女子大学
Ochanomizu University
†3 KDDI 研究所
KDDI Lab.

Vol.2015-GN-93 No.22
Vol.2015-CDS-12 No.22
Vol.2015-DCC-9 No.22
2015/1/27

2.2 WakeLock

Android OS には起動状態端末が Sleep していない状態を保持するために使用される 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，画面を点灯し続けるなどの目的で使用される．

WakeLock が行われずにディスプレイが消えた状態になると，Android 端末は Sleep 状態に入りバッテリの消費を抑える．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．

WakeLock と Sleep 状態の関係を図 1 に示す．図は端末内にで WakeLock を行うアプリケーション A および B が実行されている状態を示しており，横軸は時刻となっている．アプリケーションはそれぞれ独自のタイミングで WakeLock の取得と開放を行うことができ，取得には `aquired()` を，開放には `released()` を使用する．すべてのアプリケーションが WakeLock を開放すると端末は Sleep 状態に移行することが可能となる．

WakeLock には CPU 稼働，Screen 点灯，Keyboard の状態により，4 つの種類に分類される．表 1 に 4 種の WakeLock を示す．スクリーン状態の Dim とは薄暗い状態を示す[3].

表 1　WakeLock の種類

Flag Value	CPU	Screen	Keyboard
PARTIAL_WAKE_LOCK	On	Off	Off
SCREEN_DIM_WAKE_LOCK	On	Dim	Off
SCREEN_BRIGHT_WAKE_LOCK	On	Bright	Off
FULL_WAKE_LOCK	On	Bright	Bright

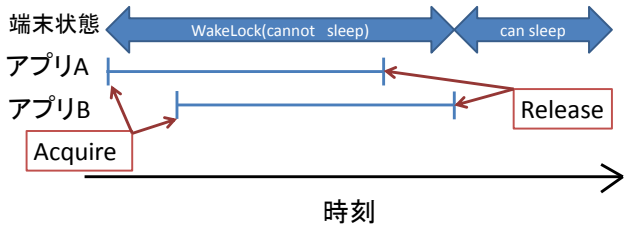

図 1　WakeLock と Sleep の関係

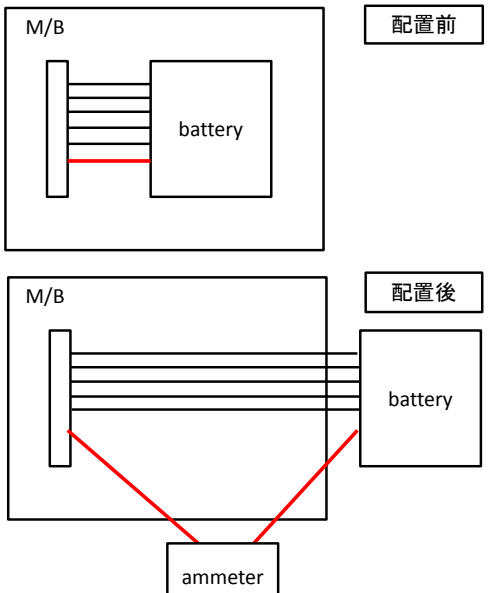

図 2　消費電力測定環境

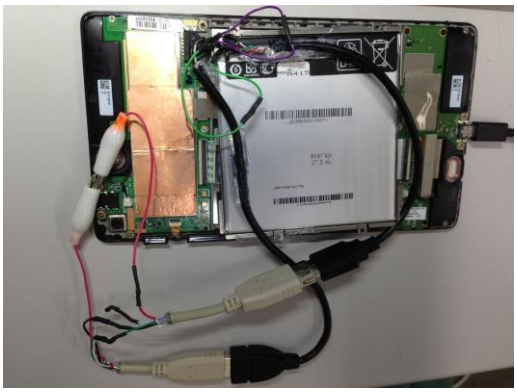

図 3　電流計測用加工済端末

3. ブロードキャストインテントによる電力消費

本章でブロードキャストインテントと電力消費の関係の調査結果について述べる．

3.1 調査環境

調査は Nexus 7 (2013)で行い，端末の OS は Android4.4, CPU は Qualcomm Snapdragon S4 Pro 1.5GHz ,メモリ 2GB である．OS にはブロードキャストインテントが発行されるたびに発行されたインテントの名前，発行プロセスの PID とプロセス名，発行時刻を記録するような修正が施されている．具体的には，`/frameworks/base/services/java/com/android/server/am/BroadcastQueue.java` 内の `enqueueParallelBroadcastLocked` メソッドおよび `enqueueOrderedBroadcastLocked` メソッドにてインテント発行時に変数 `r` 値を記録するように修正されている．

実験は特定の開発元のアプリケーション 49 個と，通信量測定用アプリケーション 1 個を用いて行った．次節以降の“noapp”は，追加でインストールしたアプリケーションが 0 個の標準状態(Android Open Source Project[4]にて配布されている OS に添付のアプリケーションのみがインストールされている状態)を，“50app”は noapp に加えて上記 50 個のアプリケーションをインストールした状態を表している．

Vol.2015-GN-93 No.22
Vol.2015-CDS-12 No.22
Vol.2015-DCC-9 No.22
2015/1/27

3.2 Android 端末の消費電力測定方法

Android 端末の消費電力を測定するために，図 2，図 3 の様に Android 端末(タブレット PC)とバッテリを導線で繋ぎ，その間にマルチメータ(MAS-345)を配置した．タブレット PC と導線およびバッテリと導線ははんだにより接続した．本稿の電流計測はすべてバッテリ残量が大きい(80%以上)状態で行い，電圧は一定であり電流と電力が比例すると仮定して，電流の測定をもって消費電力の測定とした．

3.3 ブロードキャストインテントレシーバ登録数

アプリケーションによるインテントのレシーバ登録数の調査結果を述べる．静的なレシーバ登録数の調査は，アプリケーションの仕様を記述した Android Manifest ファイルを解析し，当ファイルにてレシーバ登録を行っているアプリケーションの数を数えることにより行った．動的なレシーバ登録数の調査には，Android OS を改変し，`frameworks/base/core/java/android/content/ContextWrapper.java` 内の `registerReceiver` メソッドにて登録インテント名，登録パッケージ名を取得することにより行った．表 2 から表 4 に静的インテント登録数，動的インテント登録数，合計インテント登録数の上位 6 件を示す．

表より，静的登録は `INSTALL_REFERRER` や `APPWIDGET_UPDATE` などの，アプリケーションの状態に変更があった際に使用されるインテントのレシーブに使用されることが多いことがわかる．また，動的登録は `PROXY_CHANGE` や `SCREEN_ON` などのデバイス内の状態に変更があった場合に使用されるインテントのレシーブに使用されることが多いことがわかる．静的と動的のレシーバ登録数の合計を集計した結果を表 4 に示す．表 2 から表 4 より，静的登録と動的登録には重複が少なく，異なる傾向を示すことがわかる．

3.4 ブロードキャストインテント発行量

インストールアプリケーション数とブロードキャストインテントの発行量の関係を調査するために，Android 端末を無操作状態で 24 時間稼働させブロードキャストインテントの発行量と発行主を測定した．

図 4 に“noapp”と“50app”の 24 時間における総インテント発行量（端末全体での発行量）を，図 5 にインテントごとの発行回数（上位 10 件）を，図 6 にアプリケーション（発行主）ごとの発行回数を示す．図 6 のアプリケーション X は端末内情報取得アプリケーションであり，アプリケーション Y とアプリケーション Z はニュース配信アプリケーションである．図 4 より，インストールアプリケーション数を増やすことによりブロードキャストインテントの発行量が増加することがわかる．図 5 より，特定のインテントの発行回数が特に多いことがわかる．図 6 より，インテント発行量はアプリケーションにより大きな差があること，アプリケーション X の発行回数が非常に多いことがわかる．

表 2　静的レシーバ登録数

登録数	インテント名
34	INSTALL_REFERRER
18	APPWIDGET_UPDATE
16	c2dm. RECEIVE
11	PACKAGE_REPLACED
11	BOOT_COMPLETED
10	VIEW

表 3　動的レシーバ登録数

登録数	インテント名
34	PROXY_CHANGE
25	SCREEN_ON
25	SCREEN_OFF
16	BATTERY_CHANGED
11	CONNECTIVITY_CHANGE
11	VIEW

表 4　合計レシーバ登録数

登録数	インテント名
34	INSTALL_REFERRER
34	PROXY_CHANGE
26	SCREEN_ON
26	SCREEN_OFF
18	APPWIDGET_UPDATE
17	c2dm. RECEIVE

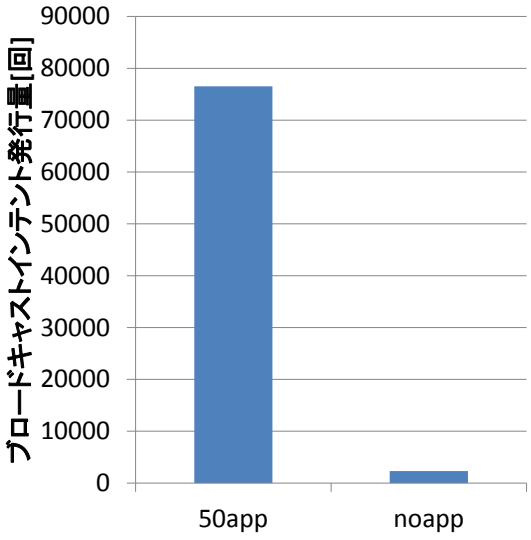

図 4　ブロードキャストインテント発行量

3.5 ブロードキャストインテント発行量が多いアプリケーション削除時の電力

前節でブロードキャストインテント発行量が特に多かったアプリケーション X を端末内から削除した場合と削除しなかった場合の電流を図 7 に示す. 電流計測は 10 分間行

Vol.2015-GN-93 No.22
Vol.2015-CDS-12 No.22
Vol.2015-DCC-9 No.22
2015/1/27

い, 無操作時の電流を電流計で測定した. 図中の“WITH_X”は端末内にアプリケーション X を入れた場合であり（これは“50app”と同一である）, “WITHOUT_X”は端末内からアプリケーション X のみを削除した場合である. 図より, アプリケーション X を削除することにより無操作時の電流が大きく減少したことを確認できる.

4. WakeLock による電力消費

本章にて WakeLock による電力消費について考察する.

4.1 計測方法

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取得した. 使用したアプリケーション数は 80 で, 計測は 24 時間(1440 分間)行った. ただし, 実験開始後 30 分はディスプレイが点灯する設定とした. Android OS には, onWakeLockReleased()メソッドの呼び出しプロセスの PID とプロセス名, 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. 十分な量の記録を行うために Logcat コマンドを起動し続けると, 同コマンドが端末の Sleep 状態を妨げるため, テキストファイルへ出力する方法を用いた.

実験に用いたアプリケーションは 2014 年 5 月 19 日の Google Play Store 無料アプリケーションランキング[5]の上位 80 件のアプリケーションである.

4.2 測定結果

図 8 に測定結果を示す. “80app”はすべてのアプリケーションを端末に導入した場合を示し, “noapp”は端末にアプリケーションを追加でインストールしていない状況（前章と同様に, Android Open Source Project にて配布されている OS に添付されているアプリケーションのみインストールされている状況）を示す. 図の“間隔平均”は WakeLock の Release が行われた間隔の平均である. 間隔が短いほど頻繁に WakeLock が行われており, 端末の Sleep が妨げられていることを意味する. 図内のバッテリ消費量は /sys/class/power_supply/battery/capacity より得られるバッテリ残量の減少量である. 両実験ともバッテリ残量が 100%の状態から実験を開始している. 測定結果より, アプリケーションを 80 個インストールし WakeLock の Release が行われる間隔が短い環境ではバッテリが 14%消費されたのに対し, アプリケーションをインストールせず WakeLock の Release が行われる間隔が長い環境

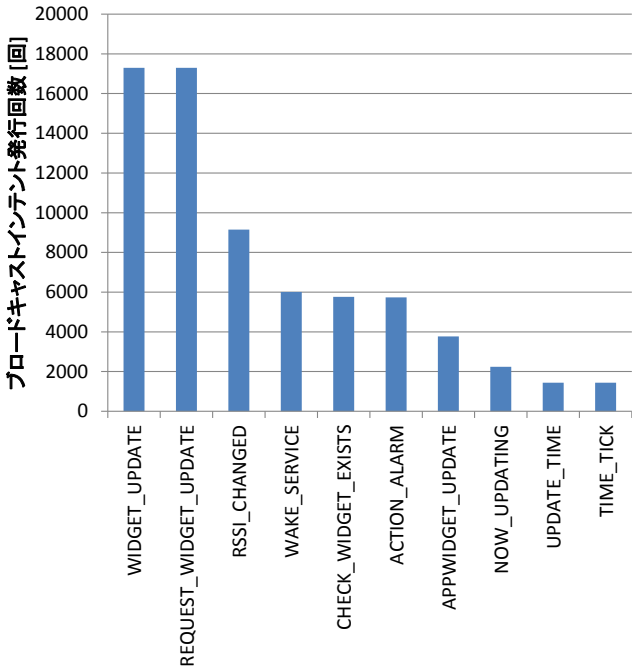

図 5 インテントごとの発行回数

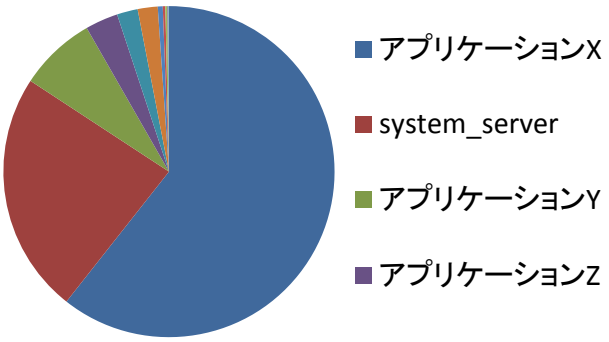

図 6 ブロードキャストインテント発行主

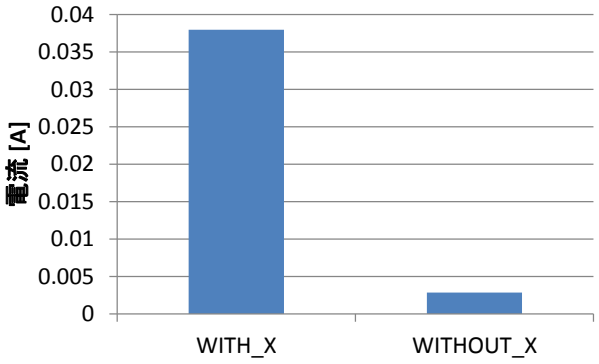

図 7 アプリケーション X の有無と消費電力の関係

ではバッテリが減少していないことがわかる. これより, WakeLock 回数と電力消費には相関があると予想すること

Vol.2015-GN-93 No.22
Vol.2015-CDS-12 No.22
Vol.2015-DCC-9 No.22
2015/1/27

ができる．

次にアプリケーションごとの WakeLock の Release 回数を調査した．Release を行ったアプリケーション（6 件）ごと．の Release 回数を図 9 に示す

4.3 WakeLock を行うアプリケーション削除時の電力

本節にて，WakeLock を行うアプリケーションのアンインストールによる省電力の効果について考察する．

すべてのアプリケーションを入れた“80app”，WakeLock を行わないアプリケーションから無作為に 6 つ選びアンインストールした“without_nowake6”，WakeLock を行う 6 件のアプリケーション（図 9 のアプリケーション A から F）を端末からアンインストールした“without_top6”，追加でインストールしたアプリケーションが 0 個の標準状態 (Android Open Source Project にて配布されている OS に添付のアプリケーションのみがインストールされている状態) である“noapp”の比較を行った．計測は 24 時間無操作でバッテリ消費量と WakeLock を Release した時刻の計測を行った．結果を図 10 に示す．図中の間隔平均は，WakeLock の Release を行った時刻とその次の WakeLock の Release を行った時刻の差の平均を示す．図より，WakeLock を行わないアプリケーションをアンインストールしても無操作時の消費電力の低減にはほとんど効果が無いが，WakeLock を行うアプリケーションのアンインストールは無操作時の消費電力の低減に大きな効果があることがわかる．すなわち，WakeLock の Release の観察により，無操作時の消費電力が多いアプリケーションの特定に成功できたと考えることができる．

各実験における WakeLockRelease 発行間隔とバッテリ残量の推移を図 11 に示す．バッテリ残量の減少速度を比較すると，“80app”と“without_nowake6”の減少速度が高く，“without_top6”が低いことがわかる．発行間隔を比較しても前者の 2 例は間隔が短く，後者は間隔が長いことがわかる．バッテリ残量は「60 秒間待機（sleep コマンド）し，バッテリ残量を調査する」ことを無限に繰り返す Shell script を動作させて測定した．よって端末が Sleep 状態に入らなければ 1 分に 1 回プロットされるが，端末が Sleep 状態に入ったため測定間隔は 1 分以上となっている．プロットの間隔が特に長い部分は，端末が長い時間 Sleep 状態であったことを意味しており．それらの時間帯では WakeLock の間隔も長くなっていることがわかる．

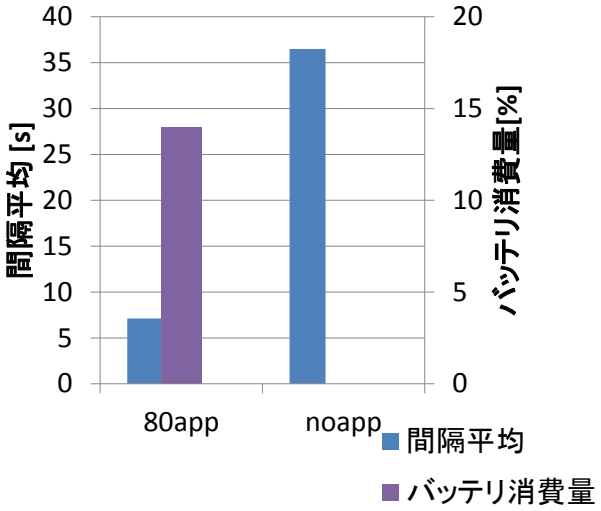

図 8　WakeLock とバッテリ消費量の関係

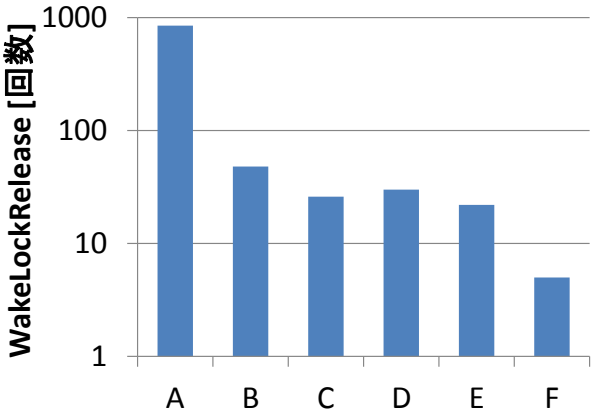

図 9　アプリケーションごとの WakeLockRelease 回数

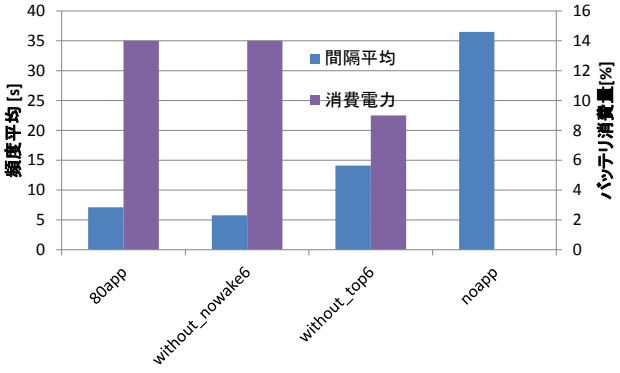

図 10　Release 回数が多いアプリケーション削除時の電力

5. 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，ブロードキャストインテントの発行回数および WakeLock の実行回数に基づく無操作時電力消費の原因となるアプリケーションの特定手法についての考察を行った．また，本手法により特定したアプリケーションを削除したところ消費電力の低減が確認され，本手法に有効性があることが確認された．

今後は，異なるアプリケーションを用いての評価を行っていく予定である．

Vol.2015-GN-93 No.22
Vol.2015-CDS-12 No.22
Vol.2015-DCC-9 No.22
2015/1/27

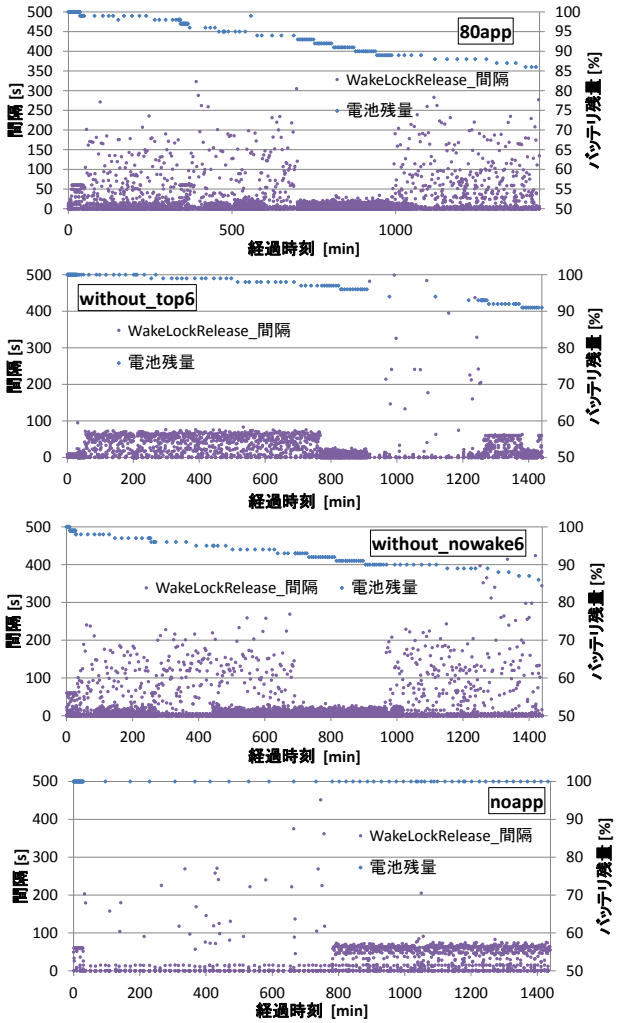

図 11　WakeLockRelease 発行間隔とバッテリ残量推移

謝辞

本研究は JSPS 科研費 24300034，25280022，26730040 の助成を受けたものである．

参考文献

1) Smartphone OS Market Share, Q3 2014.
http://www.idc.com/prodserv/smartphone-os-market-share.jsp
2) BCN
http://www.nikkei.com/article/DGXNASFK2600W_W3A320C1000000/
3) PowerManager
http://developer.android.com/reference/android/os/PowerManager.html
4) Android Open Source Project
http://source.android.com/
5) 無料トップ Android アプリ
https://play.google.com/store/apps/collection/topselling_free