Android 端末のための画面オフ状態におけるバックググラウンドタスク実行タイミング制御手法の検討

小西哲平[†1]　神山剛[†1]　川崎仁嗣[†1]　稲村浩[†1]

Android のバックグラウンドタスクには，その動作特性から実行時間においてアイドル時間が相当の割合を占めているためタスクの重畳実行による消費電力の効率化の余地がある．本研究は消費電力削減と同期干渉による BG タスクの意図しない大域起動同期への課題を指摘し，既存の Android OS のタスク管理機能の改善のためのバックググラウンドタスク実行タイミング制御方式を提案する．

Reducing the use of Energy and Wireless Resources on Android Devices during Screen Inactive Periods

TEPPEI KONISHI[†1]　TAKESHI KAMIYAMA[†1]
SATOSHI KAWASAKI[†1]　HIROSHI INAMURA[†1]

We propose a task scheduling method,　in order to improve the task management of Android OS for power efficiency as well as avoiding the global synchronization in the task invocation. We find more room for energy efficiency by concurrent execution of background tasks because the tasks have much idle and I/O wait time. We point out a phenomenon "*sync by interference*" that causes the global synchronization of task execution among Android devices.

1. はじめに

電池持ち時間が長いフィーチャーフォンに慣れた利用者にとっては，多様な利用が可能になったゆえの電池持ち時間の短いスマートフォンへの要求は厳しく，利用者の不満要因となっている[1]．さらに端末からの頻繁なネットワークアクセスにより輻輳が発生し，基地局と端末間の接続が困難になるという問題が議論されている[12]．従来のフィーチャーフォンとは違い，スマートフォンに搭載されている Android OS は利用者が操作をしない画面オフ状態でも各アプリケーション（アプリ）が独自のタイミングで端末を休眠状態から活動状態へ遷移（wakeup）させることを許容し，バックグラウンド（BG）タスクを実行させる[7]．この BG タスクの実行により，スマートフォンは画面オフ状態でも端末のエネルギーを消費することが従来のフィーチャーフォンに比べて電池持ち時間が短いことの一因であると考えられる[10][11]．さらに端末間で意図しない BG タスクの起動の同期が起こり，その無線通信の挙動により端末間で同時にタスクが起動され通信の集中が起る現象についても述べ，その対策も議論する．次節で述べるように BG タスクは実行のタイミングに任意性があり，タスクの重畳実行で消費電力を削減することが可能である．このときタスク起動が端末間で同期してしまうことで通信が集中することは考慮されなければならない．本稿では消費電力の効率化と起動タイミングの分散について要件を導出し，それらに対処する Android における BG タスクの起動タイミング制御方式を提案する．

2. BG タスクの特性と消費電力効率化の可能性

2.1 BG タスクの特性

BG タスクの起動状況を分析した結果，BG タスクの動作特性からアプリ実行時間（アプリケーションの開始から終了までの時間）においてアイドル時間が相当の割合を占めており，CPU やベースバンドチップなどのコンポーネントがアクティブな状態において必ず消費する最小限のエネルギーの部分に効率化の余地があることがわかった．

まず，画面オフ状態の BG タスクの動作特性について述べる．Android 端末から画面オフ状態における BG タスクの起動/終了時間，BG タスク動作中の CPU の稼動状況（User, System，Idle，IOWait，IRQ，SOFT IRQ）や無線送受信量を 500ms 毎に端末ログとして取得するアプリを作成し，得られた端末ログデータから，BG タスクの特性を調査した．本調査の被験者は 6 名である．ログ取得期間は 24 時間とした．

表 1 は全被験者の全 BG タスクの CPU の稼動状況の平均値を示したものである．全被験者の BG タスクの平均 CPU 使用率（System と User の合計時間の全体に占める割合）は 32%程度であった．言い換えると，BG タスクのアプリ実行時間の約 6 割が実行空き時間（Idle，I/O 待ちの合計

†1 NTT ドコモ　先進技術研究所
Research Laboratories,　NTT DOCOMO,　INC.

時間）であり，この部分の CPU リソースが利用可能であるとわかった．

次に，各コンポーネントがアクティブな状態において必ず消費する最小限の電力であるオフセットの電力消費の特徴について，電力モデル式を用いて説明する．

端末の消費電力のモデル化に関する研究[6][9]では，端末を構成する各コンポーネントで消費電力を分けて詳細な分析を可能にしているが，ここでは既存研究[6]のモデルを基本とし BG タスクが計算処理部と通信処理部の２つで構成されているとする前提を置いた単純な定式化を用いる．オフセットの考え方もこのモデルに依る．画面オフ状態における BG タスクを実行する際の端末全体の消費電力 E は式１のような電力モデル式で表せる．Ecpu と Ecpu_offset が計算処理に係る消費電力で，Ewireless と Ewireless_offset が通信処理に係る消費電力である．例えば，メモリが使用する消費電力は計算処理として Ecpu と Ecpu_offset に含まれると考える．

式 1 $E = E_{cpu} + E_{cpu\_offset} + E_{wireless} + E_{wireless\_offset}$

上記式のE_{cpu_offset}と$E_{wireless_offset}$はオフセット部分の消費電力であり，実行時間に応じて増加する．一方，それ以外の項の値は CPU の処理量や通信量に応じて増加する．つまり，idle 時間が長いということは特段の処理はなされずにE_{cpu_offset}，$E_{wireless_offset}$分の電力を消費しているということである．そこで，前述の BG タスクの特性と式１から，実行時間の短縮により消費電力の削減が出来ることがわかる．すなわち個々のアプリ独自のタイミングで単独

表 1　被験者の端末における CPU の稼動状況の平均値による BG タスクの特性

	System + User (%)	IOWait(%)	Idle(%)	IRQ + SoftIRQ (%)
全体	32	4	64	0
通信を含むバックグラウンドタスク	33	4	63	0
通信を含まないバックグラウンドタスク	29	5	66	0

表 2　検証に用いた BG タスク単体の特性

	実行時間(Tick)	CPU使用率(%)	実行空き時間率(%)
BGタスク単体	1108	28	72
	データ送信量(byte)	データ受信量(byte)	消費エネルギー(Wh)
BGタスク単体	7992	215729	0.0024

表 3　BG タスクの実行タイミングによるリソース利用結果

	実行時間(Tick)	CPU使用率(%)	実行空き時間率(%)
異なるタイミングで実行	2216	28	72
重畳実行	1112	39	61
	データ送信量(byte)	データ受信量(byte)	消費エネルギー(Wh)
異なるタイミングで実行	15984	431458	0.0049
重畳実行	13652	429267	0.0027

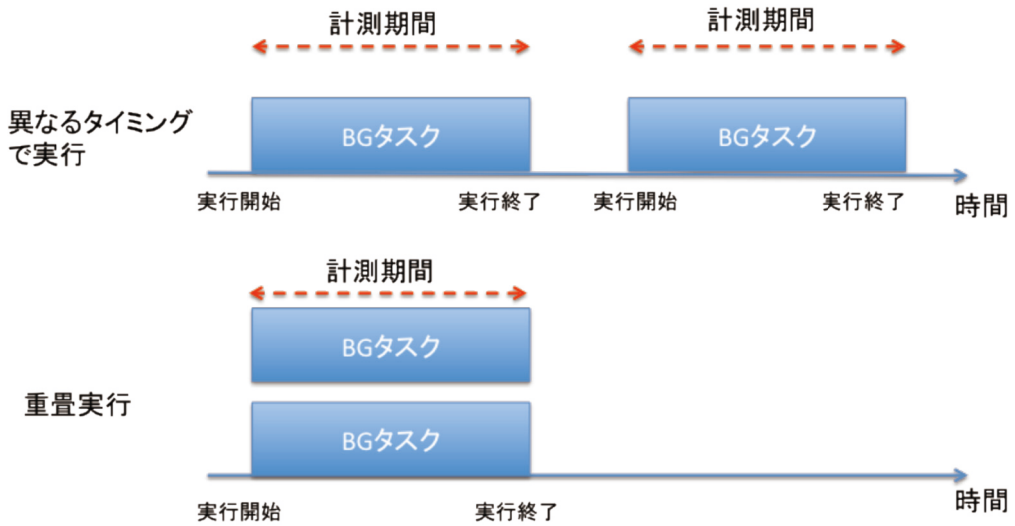


図 1　異なるタイミングで実行した場合と重畳実行した場合の計測期間

実行されていた各 BG タスクを制御し，任意のタイミングで複数同時に重畳実行し，E_{cpu_offset}，$E_{wireless\_offset}$ の消費電力項を複数のタスクで共用すれば良い[14]．画面がオフである BG タスクは利用者とのインタラクションがなく実行タイミングに任意性があるためこのような実行タイミングの調整が可能である．さらに，同時に複数の処理を行えるため，全体的な実行時間も短縮できる．

2.2 Android 端末におけるタスク制御による消費電力効率化の検証

前節で述べた BG タスクを制御し重畳実行させることで電力効率化が Android 端末において有効であることを示す．本検証のために任意のタイミングで複数の BG タスクを起動させるアプリを作成した．起動する BG タスクは，前記 BG タスクの特性から得られたものとほぼ同様の CPU・無線リソース負荷をかけるものとし，その特性は表 2 の通りである．すなわち，この BG タスクの実行開始から終了までの実行時間は 1108Tick，平均 CPU 使用率は 28%，平均実行空き時間は 72%，無線データ送信量は 7992byte，受信量は 215729byte であって，前節のＣＰＵ使用率が３割という特性に合わせた．

この BG タスクを図 1 のように二つそれぞれ異なるタイミングで実行した場合と重畳実行した場合での消費電力・CPU 利用状況・データ送受信を計測し，その変化を確認する．検証に用いた端末の仕様は以下の通り，Android OS ver. 2.3，Qualcomm 社製シングルコア CPU（クロック 1GHz），RAM 512MB，ROM 1GB，3G/WiFi 搭載のものである．通信は 3G 回線で行った．消費エネルギーの計測は，端末とバッテリを接続する端子部に電流測定器（YOKOGAWA 社製 WT1600）と安定化電源を接続して行った．BG タスクをそれぞれ異なるタイミングで実行する場合は，図 1 のように二回分の BG タスクの実行開始から終了まで計測した値を用いる．重畳実行した場合は，二つのタスクを同時に実行させ，実行開始から実行終了まで計測した値を用いる．この計測をそれぞれ 10 回行い，平均値を求めた．

実験結果は表 3 の通りである．重畳実行した場合の実行時間は 1112Tick であり，異なるタイミングで実行した場合の実行時間は 2216Tick であることから，それぞれ異なるタイミングで実行した場合の約半分の実行時間で完了したことが分かる．また，CPU 稼動状況については単体で実行した場合に比べて，実行空き時間率が約 10%減少し，CPU 使用率が 10%増加した．このことから，実行空き時間を他の処理で充填出来ていることと，それによって実行時間が短縮されていることが確認できた．二つの BG タスクを重畳実行した場合，消費電力はそれぞれ異なるタイミングで実行した場合と比較し，約 54%に削減されていた．これは，前節で述べたように，重畳実行することで，BG タスクの実行空き時間を利用し，実行時間を短縮することで，式 1 の実行時間に応じて増加するE_{cpu_offset}と$E_{wireless_offset}$を共用し，消費電力を削減出来たと考えられる．以上のことから，BG タスクを重畳実行させることで電力効率化が有効であることを示された．

3. Android OS の BG タスク管理機構

Android OS のタスク管理機能において前記の BG タスク制御による消費電力削減手法の適用を考える．本節では以降の議論に必要な Android OS のタスク制御機構の説明を行う．BG タスクの実行契機である AlarmManager（AM）と BroadcastIntent（BI）機構の動作について述べる．

3.1 AlarmManager

AlarmManager(AM)とは，Android OS が提供する実行管理機構の一つで，BG タスクの実行を一元的に管理している．AM は BG タスクを実行時間や繰り返し間隔などの実行条件に従って起動・管理するスケジューラである．開発者は AM が提供するメソッドを使用し，各アプリの BG タスク実行時間と時間指定タイプによる実行条件をスケジューラエントリとして登録する．

3.1.1 時間指定タイプ

時間指定タイプは以下の四つがある．

- RTC
  指定時間を RealTimeClock(RTC)に基づく実時間で指定する．AM に指定した時間を指定時間とする．端末を wakeup しないため，この指定時間以降に端末が何らかの契機により wakeup していればその時点で BG タスクを実行する[8].
- RTC_WAKEUP
  指定時間を RTC に基づく実時間で指定する．指定時間に端末を wakeup させ BG タスクを実行する．端末が wakeup する時間を指定することを本稿では Wakeup 指定と呼ぶ.
- ELAPSED_REALTIME
  指定時間を端末の電源が入った（Boot）時からの経過時間で指定する．指定時間以降に端末が wakeup した時点で BG タスクを実行する．
- ELAPSED_REALTIME_WAKEUP
  指定時間を Boot 時からの経過時間で指定する．指定時間に端末を wakeup させ，BG タスクを実行する．RTC 型と ELAPSED 型の違いは時刻の基準のみであり，差はあまり大きくないが，例えば目覚まし時計アプリを作るには RTC_WAKEUP 指定とするのが直裁である.これに対して，「端末起動時刻から３０分毎」のような指定には ELAPSED 型が便利である．この指定を活用すると端末間の起動集中を回避する動作が指定しやすいため価値が大きい．ELAPSED 型では時刻

基準が端末毎に異なるためである．

3.1.2 メソッド

BG タスクを AM に登録するメソッドは以下の三つである．

- Set
  一回限りの実行を，実行時間を指定することで行う．
- SetRepeat
  同一のタスクの繰り返し起動を行う.初回実行時間と実行間隔を指定する．
- SetInexactRepeat(SIR)
  初回実行時間と実行間隔を指定する．Android OS バージョンが 2.3.3 の場合では実行間隔が 15 分, 30 分, 1 時間，半日，一日で指定された BG タスクは重畳実行の対象となり， 15 分単位の周期タイマにまとめて登録される．OS バージョンが 4.0.3 以上の場合では，実行間隔が 15 分の倍数の BG タスクが重畳実行の対象となる．

3.2 BroadcastIntent

BroadcastIntent(BI)とは電池残量やネットワーク状態の変化など，システム全体に関わる情報を全てのアプリに発信する機能で，各アプリが BI からの情報を受信することで，端末の状態変化に応じた処理を実装することができる[13]．BI を発行する際に，端末は wakeup する．

3.1 節で述べたように，AM に登録された wakeup 指定なしの BG タスク（RTC，ELAPSED）は指定時間以降に wakeup したタイミングで実行される．従って，BI 発行のための wakeup であっても AM にエントリされた BG タスクの実行契機となり得るため，課題分析の際は BI を考慮する必要がある．

4. BG タスク制御での端末 wakeup 回数の削減

BG タスク実行契機に基づき，端末の画面オフ状態の消費電力をモデル化する．これに被験者による実際の利用状況を想定したデータを分析して得られた実数を当てはめることで支配的なパラメータを見出す．その結果消費電力は BG タスクの実行を伴う wakeup の回数が支配的なパラメータである事が分かった．

4.1 画面オフ状態の消費電力のモデル化

3 章で見てきたように画面オフ状態における BG タスク実行は AM 契機，BI 契機に分けられる．これを基に一時間当たりの BG タスクの消費電力 E をモデル化する．AM 契機で wakeup した際に BG タスクが実行された回数を W_{a1}, BI 契機で wakeup した際に BG タスクが実行された回数を W_{a2} とする．また，BG タスクの実行による消費電力を E_a とする． BI 契機で wakeup した際に BG タスク実行されなかった回数，つまり BI 発行のみが行われた回数を W_b, その時の消費電力を E_b とする．これらを用いると E は式 2 で表せる．

式 2 $$E = E_a * (W_{a1} + W_{a2}) + E_b * W_b$$

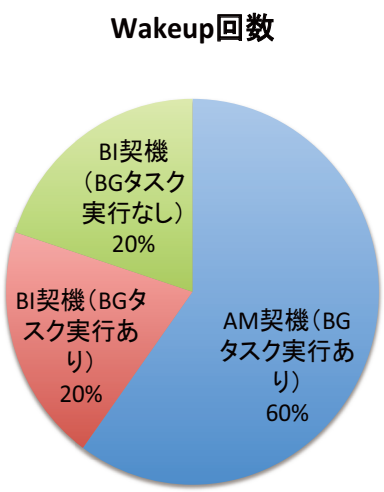
$W_{a1} + W_{a2}$ を W_a と表すと，式 2 は式 3 で表せる．

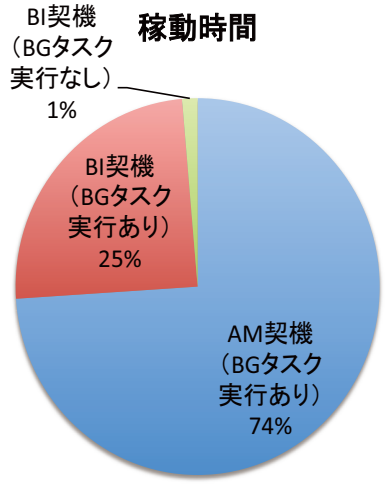
式 3 $$E = E_a * W_a + E_b * W_b$$

E_a は計算処理に係る電力 P_{cpu} と計算処理の際に必要な電力（計算処理のオフセット）P_{cpu_offset}，無線通信処理に係る電力 $P_{wireless}$ ，無線通信処理の際に必要な電力（無線通信処理のオフセット）$P_{wireless\_offset}$，BG タスクの稼動時間 T_a によって式 3 は式 4 で表される．式 4

$$E_a = (P_{cpu\_offset} + P_{wireless\_offset} + P_{cpu} + P_{wireless}) * T_a$$

表 4 BG タスクの実行契機ごとの wakeup 回数と平均稼働時間





	AM契機（BGタスク実行あり）	BI契機（BGタスク実行あり）	BI契機（BGタスク実行なし）
Wakeup回数	12	4	4
平均稼動時間（秒）	9	9	0.5未満

Wakeup 指定なしの BG タスクは wakeup した際に複数重畳実行されるため, 各W_aで平均 n 個の BG タスクを重畳実行しているとすると，E_aは式 5 で表せる．

式 5

$$E_a = \left(P_{cpu_{offset}} + P_{wireless_{offset}} + n\left(P_{cpu} + P_{wireless}\right)\right) * T_a$$

計算処理と無線通信のオフセット$(P_{cpu_{offset}} + P_{wireless_{offset}})$に n が積算されていないのは，重畳実行の際にオフセットは共通利用可能であり，処理量分$(P_{cpu} + P_{wireless})$ だけ電力が増加するためである．Eb は BI 発行の時間 Tb を用いて式 6 で表される．

式 6 $$E_b=(P_{cpu_{offset}} + P_{cpu}) * T_b$$

一時間当たりの AM による BG タスクの総登録数を Na とすると$N_a = n * W_a$ であることから，$n = N_a/W_a$ を式 5 に代入すると，E は式 7 となる．

式 7 $$E = (P_{cpu_{offset}} + P_{wireless_{offset}}) * T_a * W_a + N_a * \left(P_{cpu} + P_{wireless}\right) * T_a + (P_{cpu_{offset}} + P_{cpu}) * T_b * W_b$$

4.2 実際の利用状況を想定したパラメータの見極め

4.1 節のモデル式について，支配的なパラメータを見つける．実利用を想定した検証を行うため検証用の端末を利用する被験者を約 50 人募り，アプリ毎の稼動時間，稼動回数を記録するアプリを被験者の端末にインストールして調査を行った．検証に用いた端末の仕様は Android OS ver. 2.3, Qualcomm 社製シングルコア CPU（クロック 1GHz），RAM 512MB，ROM 1GB，3G/WiFi 搭載である．調査期間は一週間とした．BG タスクとして動作するアプリはユニークユーザ数(ユーザ毎に一回以上稼動させたことで計数する)の多いものを対象として検討したところ出荷時インストール状態のアプリでカバーされた．そこで，出荷時インストール状態の端末の BG タスク実行ログ（wakeup 回数，稼動時間，指定時間/間隔情報，通信の有無で構成される）を取得し，モデル式のパラメータ値を推定した．一時間当たりの平均値を表 4 に示す．

Ta が平均 9 秒， Tb-が平均 0.5 秒未満であり，Ta が Tb の約 18 倍も大きいこと, Na が定数であることから, E は BG タスク実行契機となる wakeup 回数 Wa が支配的なパラメータである事が分かる．したがって，BG タスク制御により Wa を制御することが消費電力削減に有効である．

5. Android OS の BG タスク管理における課題

既存の Android OS の AM には Wa の削減余地がある．さらに通信集中の観点から実時間指定の同期干渉による BG タスクの起動同期への対処も望まれる．

5.1 Wa の削減余地

既存の Android OS の AM では，BI 契機の wakeup によ

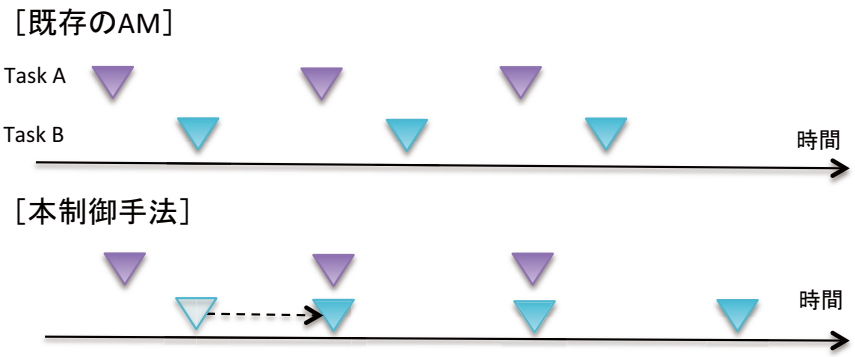


図 2 STRICT_PERIOD の動作例

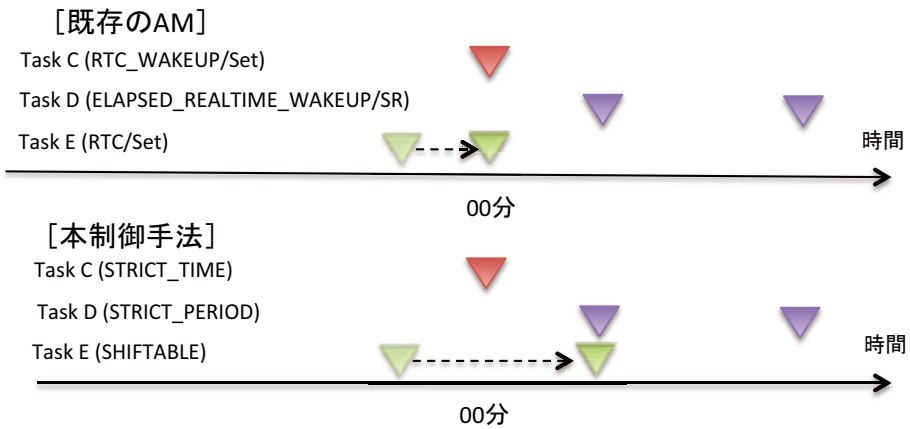


図 3 SHIFTABLE の動作例

って wakeup 指定なしの BG タスクが実行されることで，BG タスクが重畳されることなく散漫に実行されるため，W_{a2} が増加する．AM には BG タスクを重畳実行するための SetInexactRepeating(SIR)という仕組みが導入されているものの，周期が 15 分の倍数のものしか重畳実行の対象とはならない．

5.2 実時間指定と同期干渉による BG タスクの起動同期

既存の AM では API によって BG タスクの起動時間を実時間で指定可能であることから，配慮のないプログラミングや運用により特定実時間に同期して多くの端末で通信を伴う BG タスクが実行されることで輻輳が発生する可能性がある．更に，通信を伴わない BG タスク(task A)であっても端末に wakeup を掛けることで，それまで未実行であった wakeup 指定なしの BG タスク(task B)が実行される．この task A が定時に起動するものであるとし，task B が通信を行うものであったとき，この 2 者の連携により定時において端末間で通信の同期実行が起る．この task A と task B の連携に見られるタスク間の同期実行を本論文では「同期干渉」と定義する．例えば通信を行わない目覚し時計アプリによって端末が wakeup されることによって，通信を行なうアプリが起動されると，同じ時刻に目覚しを設定した端末の間で同期した通信が意図せずに発生してしまう．この例では目覚し時計アプリが他アプリへの同期の干渉源として機能している．

6. BG タスク制御方法の要件

本研究では以下の三つを BG タスク制御方法の要件として定める．

1. 端末毎の通信タイミングを分散するために実時間で起動するタスクを管理する
2. 既存アプリの再設計を回避する
3. アプリの実時間性を確保する

要件 1 は，通信集中の発生を回避するためである．BG タスクの実行タイミングの制御によって複数の端末間の通信タイミングが同期すると通信集中が発生し，ネットワーク（NW）負荷が問題となる．要件 2 は，開発者の負担をなるべく減らすためである．制御手法を導入するために既存のアプリを改変することはアプリ開発者にとって開発工数，費用の面から負担となる．要件 3 は，アプリのサービス価値を保つためである．目覚まし時計アプリやリマインダアプリ，サーバとのセッション保持（KeepAlive）を行うタスクなどの実時間性が必要なものは，実行時間や実行間隔がずれるとサービス価値を失う．

7. 関連研究/関連技術

Android OS において，W_a を削減することで電力を削減する既存研究として，周期的な BG タスクの初回実行時間をシフトし，重畳実行するタスク数を増やす Batch Scheduling[3]という手法が提案されている．これは BG タスクの周期に着目し，BG タスクの初回実行時間をなるべく重畳実行できるようシフトし，W_a を最小化する手法である．しかし，周期性のない実時間指定の BG タスクは制御対象でなく開発者が利用できるため，要件 1 は満たさない．また，BG タスクの指定時間は無視して実行時間をシフトするので要件 3 を満たさない．指定されたタスクを自動的に停止するタスクマネージャアプリが利用者によるアドオンで用いられることがある．一例としてタスクキラー[8]が挙げられる．このようなアプリを用いてＢＧタスクの停止による電力削減の試みが利用者によって行われている．タスクの停止によって W_a を削減できるが，アプリの実時間性は守られないため要件 3 を満たさない．更に，タスクキラーは実行中のタスクを監視し続けるため，タスクキラー自体が電力を消費する．

なお，iOS[5]や Windows Phone[6]においては BG タスクの起動の指定や利用に制約がある．iOS は，VoIP や音楽再生など特定のアプリカテゴリのものしか画面オフ状態では実行できない．Windows Phone は BG タスクの使用するリソースを管理し制約するためタスクの起動時間の遅延や未実行が起こることを予め許している．

8. BG タスク実行タイミング制御手法の提案

AM の指定方法からアプリ開発者が意図した BG タスクの時間制約が明らかになるよう AM を再整理，新規の意味論を追加した上で，Android OS 側で自動的に BG タスクを重畳実行することで，既存アプリの挙動に影響を与えず W_a を削減する手法を提案する．我々の提案は，BI 契機の wakeup によるタスク起動において異なる実時間特性のものを分離し，同時実行させないことと，API の意味論の整理，追加と併せて特性の同じタスクの重畳実行の可能性を高めるものである．同期干渉によるタスク起動の同期の可能性を抑え，管理するべく考慮されている．

8.1 BG タスクの時間制約と AM の指定方法の関係

アプリの挙動に影響を与えず W_a を削減するためには，アプリ開発者の意図した BG タスクの時間制約を理解した上で，BG タスクを制御する必要がある．しかし，既存の AM の指定方法は，表 5 のように様々な組み合わせがあり，指定方法からだけではアプリ開発者の意図した BG タスクの時間制約が明確ではない．例えば，一定の実行間隔で実行する KeepAlive タスクは実時間である必要は無いため ELAPSED_ REALTIME_WAKEUP にて指定し SIR API を利用すべきであるが，RTC の実時間指定をはじめ，複数の指定方法で実現可能である．BG タスクをその実行にお

ける時間制約の観点から見ると以下のように三つに分類できる．

・ 時間厳守タスク
実行時間が実時間でなければならない BG タスクである．目覚まし時計やリマインダアプリのように，実時間で動作しなければサービス価値を失う BG タスクがこれに当たる．

・ 間隔厳守タスク
実行時間は実時間で無くてもよいが実行間隔を守らなければならないタスクである．KeepAlive やデータロギングなど一定の実行間隔を守らなければサービス価値を失う BG タスクがこれに当たる．

・ 実時間性制約のないタスク
実行時間指定に制約が少ない BG タスクである．例えば，少なくとも一日に一回サーバとデータ更新が出来れば良いような BG タスクがこれに当たる．実行時間に指定はなく，実行間隔への要求も任意性が高い．

そこで，既存の AM の指定の意味論を反映し，BG タスクの時間制約と指定方法を 1 対 1 対応にする．これにより，アプリ開発者の意図した BG タスクの時間制約を AM の指定方法から判断できるようになる．

8.2 AM の API の意味の整理と制御手法

BG タスクの時間制約と AM の指定方法が 1 対 1 になるように，AM を再設計する．

・ STRICT_TIME（時間厳守タスク）
指定時間に端末を wakeup して BG タスクを実行する．従来の Set/SetRepeat のうち初回実行の指定時間が実時間であるものをここにまとめる．

・ STRICT_PERIOD（間隔厳守タスク）
指定された間隔毎に端末を wakeup して BG タスクを実行する．アプリからは初回実行時間は指定できず，Batch Scheduling[3]のように，最も重畳実行ができる時間に初回実行する．図 2 に STRICT_PERIOD の動作例を示す．STRICT_PERIOD の BG タスクが新たに登録された場合(Task B)，既に登録されている STRICT_PERIOD(Task A)の周期を確認し，同周期であれば，その時点まで初回実行を遅延する．同周期のものが無ければ，次に実行される STRICT_PERIOD の実行時間を初回実行時間として指定する．登録されている STRICT_PERIOD が無い場合は，TaskB が登録された時間に初回実行する．

・ SHIFTABLE (実時間性制約のないタスク)
充電中かどうかに関わらず，画面オフ状態では，図 3 のように STRICT_PERIOD による wakeup 契機でのみ実行する．画面オン状態では，指定時間通りに実行する. STRICT_TIME による wakeup ではこのクラスのタスクは重畳実行を行わない．

8.3 Wa 削減の仕組みと効果

SHIFTABLE 指定のタスクは BI 契機による wakeup では実行されないため，モデル式 2 の W_{a2} をゼロにする事が出来る．これにより，表 4 の実利用データから，一時間当たり 4 回の wakeup 削減効果が期待できる．また，STRICT_PERIOD で指定されたタスクも SIR とは違い，周期が 15 分未満の BG タスクも重畳実行対象とするため SIR を用いる場合より W_{a1} を削減できる可能性が高い．STRICT_TIME による wakeup では，このクラスのタスクは重畳実行を行わないことで BG タスク起動が実時間に同期することを回避する．

表 5 AM の指定方法

	時間指定タイプ	メソッド	実時間性制約がある		実時間性制約がない
			時間厳守	間隔厳守	
従来の指定方法	RTC_WAKEUP	Set	○	○	
		SR	○	○	
		SIR		○	
	ELAPSED_REALTIME_WAKEUP	Set	○	○	
		SR	○	○	
		SIR		○	
	RTC	Set			○
		SR			○
		SIR			○
	ELAPSED	Set			○
		SR			○
		SIR			○
本提案手法の指定方法	STRICT_TIME		○		
	STRICT_PERIOD			○	
	SHIFTABLE				○

8.4 制御手法と要件の関係

要件 1 との適合性について述べる．STRICT_PERIOD の初回実行時間は端末毎に任意である，即ちランダムに決めることもできるため，端末毎の実行タイミングは分散できる．図 3 のように実時間で実行する STRICT_TIME と SHIFTABLE との重畳実行を避けることで，Wakeup 指定なしの BG タスクに対する同期干渉による起動集中を回避可能である．開発者の指定により STRICT_TIME の BG タスクが通信を伴う場合，通信集中の発生自体は回避できないが，その原因となる BG タスクの識別は容易である．STRICT_TIME のみが実時間実行可能なように AM を整理しているため，STRICT_TIME 指定されているかどうかを確認するだけでよい．問題が発生した場合はこの識別手段によって容易に対処できる．要件 2 については，本提案は既存の AM の意味論の再整理であるため，既存 API の動作は本提案の枠組みに含まれており満たすことができる．既存の API 利用のままでは本提案の効果は限定的であるが従来通り動作可能である．要件 3 については，STRICT_TIME/PERIOD の使用により，時間厳守タスク，間隔厳守タスクの時間制約を保持したまま実行できる．

9. 結論

Android のバックグラウンド(BG)タスクには，その動作特性から実行時間においてアイドル時間が相当の割合を占めているため消費電力において効率化の余地がある．BG タスクを制御し重畳実行させることで消費電力の効率化が Android 端末において有効であることを示した．Android OS のタスク管理機能において前記の BG タスク制御による消費電力削減手法の適用を検討した．端末の画面オフ状態の消費電力をモデル化し，被験者による実際の利用状況を想定したデータを分析して BG タスク実行契機に基づく支配的なパラメータを見出した．その結果，BG タスクにおける消費電力においては，タスク実行を伴う wakeup の回数が支配的なパラメータであった．本研究ではこれらを明らかにした上で，消費電力削減と，通信集中の観点からの同期干渉による BG タスクの起動同期への課題を指摘し，既存の Android OS のタスク管理機能の改善のための BG タスク実行タイミング制御方式を提案した．我々の提案は，BI 契機の wakeup によるタスク起動において異なる実時間特性のものを分離し，同時実行させないことと，API の意味論の整理，追加と併せて特性が同じタスクの重畳実行の可能性を高めるものである．この分離により同期干渉によるタスク起動の同期の可能性を抑えるべく考慮されている．今後は本提案の効果について評価を進めていきたい．

謝辞 本論文の作成にご協力頂いた NTT ドコモ先進技術研究所の同僚である森岡康史氏と大久保信三氏に感謝する．

参考文献

1) MMD 研究所， 2012 年夏発売の携帯電話・スマートフォン新端末購入意欲調査
http://prtimes.jp/data/corp/4240/37fd0cd16be16277bb031bbb4a89494c.pdf
2) M. Calder and M. Marina. Batch scheduling of recurrent applications for energy savings on mobile phones. In Proc. of 7th Annual IEEE Communications Society Conference on Sensor Mesh and Ad Hoc Communications and Networks, SECON 2010.
3) F. Qian， Z. Wang， A. Gerber， Z. Mao， S. Sen， and O. Spatscheck. Profiling Resource Usage for Mobile Applications: a Cross-layer Approach. In Proc. of Mobisys， 2011.
4) Local Notification および Push Notification プログラミングガイド
https://developer.apple.com/jp/devcenter/ios/library/documentation/RemoteNotificationsPG.pdf
5) Windows Phone のバックグラウンドエージェントの概要
http://msdn.microsoft.com/ja-jp/library/hh202942(v=vs.92).aspx
6) Yusuke KANEDA，Takumi OKUHIRA，Tohru ISHIHARA，Kenji HISAZUMI， Takeshi KAMIYAMA and Masaji KATAGIRI， 2010， A Run-Time Power Analysis Method using OS-Observable Parameters for Mobile Terminals. In Proc. of Int'l Conf. on Embedded Systems and Intelligent Technology 2010.
7) Alarm Manager
http://developer.android.com/intl/ja/reference/android/app/AlarmManager.html
8) Advanced task killer. https://market.android.com/details?id=com.rechild.advancedtaskkiller
9) P. Bellasi， W. Fornaciari， and D. Siorpaes， ”Predictive Models for Multimedia Applications Power Consumption based on Use-Case and OS Level Analysis. In Proc. of DATE 2009.
10) J. Sharkey. Coding For Battery Life.
http://code.google.com/events/io/2009/sessions/CodingLifeBatteryLife.html， 2009.
11) Optimizing Battery Life,
http://developer.android.com/training/monitoring-device-state/index.html
12) Nokia Siemens Networks Smart Labs-Understanding Smartphone Behavior in the Network White Paper
13) BroadcastIntent,
http://developer.android.com/reference/android/content/Context.html#sendBroadcast(android.content.Intent)
14) V. Kon onen and P. Paakkonen: Optimizing power consumption of always-on applications based on timer alignment. In Proc. of COMSNETS 2011.