情報処理学会論文誌 Vol.56 No.3 1121–1131 (Mar. 2015)

推薦論文

アプリごとのトラヒックとユーザの利用状況を考慮したスマートフォンの通信制御手法

高木 雅[1,a)] 川原 圭博[1,b)] 浅見 徹[1,c)]

受付日 2014年7月11日, 採録日 2014年12月3日

概要：本稿では，スマートフォンにインストールされている様々なアプリについて，アプリごとに異なるトラヒックの性質とユーザの利用状況を考慮し，アプリ単位での通信制御を行うことで，携帯電話網を流れるデータトラヒックと制御信号，端末の消費電力を効果的に削減する手法を提案する．我々の調査によると，1 台のスマートフォンには平均で 58 個のアプリがインストールされており，アプリによってトラヒックの性質やユーザの利用頻度は大きく異なる．なかには，ユーザがまったく利用しなくても定期的に通信を行うアプリが存在し，データトラヒックと制御信号を合わせたトラヒック量や消費電力に影響を与えている．我々の手法は，アプリ起動やウィジェット利用の有無などの情報に基づいて不要アプリを判別し，それらの通信を選択的に遮断することで，データトラヒック自体を効果的に削減する．また，必要アプリについては，各アプリのトラヒックの統計的な性質に基づいてアイドルタイマを動的に設定することで，制御信号量と消費電力を削減する．さらに，タッチパネル操作やボタン操作と通信の発生タイミングには強い関連があるため，ユーザ操作からの経過時間も考慮して，アイドルタイマを調整する．Android SDK には，本手法で必須となるアプリ単位での通信制御の API が存在しない．そこで我々は，Android が Linux カーネル上で動作するミドルウェアであることに着目し，`iptables` コマンドを用いて制御機構部分を実装した．さらに，評価実験において，制御信号を最大 86.1%，消費電力を最大 37.6%，同時に削減できることを確認した．

キーワード：通信制御，携帯電話網，制御信号削減，省電力化，統計

A Smartphone Traffic Control Based on App-by-App Traffic and Usage

Masaru Takagi[1,a)] Yoshihiro Kawahara[1,b)] Tohru Asami[1,c)]

Received: July 11, 2014, Accepted: December 3, 2014

Abstract: In this paper, we propose a traffic control that individually deals each app installed on the smartphone not only to efficiently reduce data traffic and signaling on the mobile network, but also to save power of the device, considering the traffic characteristic and the usage of each app. According to our research result, there is an average of 58 apps installed on a single smartphone and their traffic characteristic and usage differ substantially with each other. Some apps go so far as to generate periodical traffic without any user operation. In our proposal, we reduce the data traffic efficiently by selectively intercepting the traffic from/to specific apps, which have never been launched by user. In addition, we save the signaling and power consumption by adjusting the idle timer value of mobile network interface. Moreover, we take account of the relationship between the user operation on the touch panel and the traffic occurrence. Although the function to control the traffic of each app is important in our method, Android SDK does not provide such functions. Hence we realized the traffic control function using the `iptables` command included in the Linux kernel on which Android platform runs. Our method can reduce up to 86.1% of signaling and up to 37.6% of power consumption at a time.

Keywords: Traffic Control, Mobile Network, Signaling Reduction, Power Saving, Statistics

1. はじめに

近年の携帯電話システムは，大きく 3 つの問題に直面している．1 つ目の問題は，スマートフォンの普及にともなってデータトラヒックが急増し，都市部を中心に実効スループットが低下していることである．スマートフォンのユーザはフィーチャーフォンのユーザと比較して，平均で 10 倍以上のデータトラヒックを発生させるといわれており，エリクソンの報告によると，ここ数年間，携帯電話網を流れるデータトラヒックは年間 2 倍のペースで指数関数的に増加を続けてきた [3]．今後もデータトラヒックは同様のペースで増加を続けると予想されており，2020 年までに 2013 年実績の 100 倍に達する計算になる．実効スループットの低下はユーザの体感品質を大きく低下させるため，携帯電話事業者にとって実効スループットの改善は急務である．対策として，基地局の小セル化による単位面積あたりの収容能力の向上や，Wi-Fi スポット整備による固定網へのオフロードを促進しているものの，基地局の新設には計画から運用開始まで半年から 1 年という長い時間が必要であり，Wi-Fi でのオフロードにはカバーできるエリアや効果のあるユーザ層が限定されるという課題がある．

2 つ目の問題は，データトラヒックの急増にともなって制御信号が急増し，コアネットワークの処理能力を瞬間的に上回る事態が発生していることである．常時のインターネット接続を前提とするスマートフォンは，フィーチャーフォンより高頻度で通信を行い，より多くの制御信号を発生させることが知られている．実際に，スマートフォンの普及にともなって，2009 年から 2012 年までの 3 年間で制御信号は 5 倍に増加している [4]．2012 年には，国内最大手の携帯電話事業者である NTT ドコモのネットワークにおいて，制御信号の量がコアネットワークの処理能力を上回ったことによる大規模な通信障害が発生している [5]．ひとたび通信障害が発生すると，数百万人のユーザが携帯電話サービスを利用できなくなるため，早急な対策が求められる．

3 つ目の問題は，スマートフォンのバッテリ駆動時間がフィーチャーフォンと比べて著しく短いことである．フィーチャーフォンが 1 回の充電で 1 週間使えることも珍しくないのに対し，ほとんどのスマートフォンは 2 日に 1 度以上の頻度で充電する必要がある．さらに，フィーチャーフォンの平均的なバッテリ容量が 1,000 mAh 未満，スマートフォンの平均的なバッテリ容量が 2,000 mAh 台であることをふまえると，消費電力の差には 10 倍近い開きがある．Carroll らによると，スマートフォンの消費電力の約 85%は，ディスプレイ，通信モジュール（LTE，3G，Wi-Fi），CPU の 3 点が占めており，通信モジュールの消費電力はトラヒックや制御信号の量と密接な関係にある [6]．

本稿では，スマートフォン上で利用可能な様々な情報を活用し，スマートフォン上で，ユーザに不要な通信を間引いたり，アイドルタイマを動的に設定したりすることで，携帯電話網を流れるデータトラヒックと制御信号，端末の消費電力を効果的に削減する手法を提案する．具体的には，アプリの起動頻度，通知の頻度，ウィジェットの利用状況，画面の点灯状態，実行中のアプリ，各アプリの通信間隔の分布，ユーザ操作からの経過時間，といった情報を活用して LTE/3G 接続時に適切な通信制御を行う．

本稿の構成は以下のようになっている．まず，2 章では実際に稼働中の Android 端末から収集したトラヒックの統計的な性質や，通信中のスマートフォンの消費電力の性質を紹介する．次に，3 章で本稿の提案手法であるアプリ単位での通信制御について詳しく述べた後，4 章で提案手法の性能評価を行う．さらに，5 章で関連研究について触れ，6 章で本稿の内容をまとめる．

2. スマートフォンの実際の利用状況

2.1 調査方法

スマートフォンが生み出すトラヒックの性質は，インストールされているアプリの組合せに大きく依存する．2014 年現在，Google PLAY では 100 万本以上のアプリが公開されており，それぞれ発生させるトラヒックの量や頻度が大きく異なると考えられる．先行研究でも，実ユーザのトラヒックを調査した例は存在するが，我々が着目している，アプリごとのトラヒックの性質の違いを明らかにするような公開データの存在は確認できなかった [7], [8], [9], [10]．そこで我々は，その実態を調査するため，トラヒックの統計データを収集する Android アプリ「通信量お知らせアプリ」を製作し，2013 年 4 月より Google PLAY で公開して協力者を募ってきた [11]．2014 年 6 月現在，1,800 ユーザ，のべ 78,200 アプリ分のデータを収集しており，ユーザ数は現在も増加を続けている．この「通信量お知らせアプリ」は，

- インストール済みアプリの名称とパッケージ名
- 各アプリの累計通信量（送受信，画面状態別）
- 各アプリの累計通信回数（送受信，画面状態別）
- 各アプリの通信間隔の分布（送受信，画面状態別）
- 通信発生時の各アプリの実行状態

といった統計情報を記録し，週に 1 度，我々のサーバへ

[1] 東京大学大学院情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo, Bunkyo, Tokyo 113–8656, Japan
a) takagi@akg.t.u-tokyo.ac.jp
b) kawahara@akg.t.u-tokyo.ac.jp
c) asami@akg.t.u-tokyo.ac.jp

本論文の内容は 2013 年 7 月のマルチメディア，分散，協調とモバイル（DICOMO2013）シンポジウム 2013 にて報告され，同研究会主査により情報処理学会論文誌ジャーナルへの掲載が推薦された論文である．

送信する．なお，データの重複管理に必要となる端末固有ID，機種名，OSバージョンの情報を除いては，ユーザに関する情報はいっさい収集していない．それゆえ，ユーザの年齢層や性別などは不明である．ただし，Google PLAYの統計情報によると，ユーザの96.87%は日本国内におり，通信キャリア別では，NTTドコモが58.9%，auが17.7%，SoftBankが7.5%を占めている[11]．この比率は，Google PLAYにおける各通信キャリアのシェア比率（4.4%，1.4%，0.5%）とほぼ一致している．

2.2 トラヒック

2.2.1 全体の概要

図1は，各ユーザのスマートフォンにインストールされていたアプリ数の分布である．1,800ユーザでの平均は58個，最大は629個であり，一般的なユーザのスマートフォンには，40〜50個のアプリがインストールされていると考えて差し支えない．

表1は，収集した全ユーザの合計のトラヒックにおいて，コネクション数の多い上位16アプリとそのユーザ数を示した表である．ここでは「コネクション」という単語を便宜上「途中に1秒以上の間隔のない一続きの通信」という意味で用いており，TCPにおける「コネクション」とは定義が異なる．表1において特筆すべきは，100万本以上あるアプリの中で，コネクション数の分布が特定のアプリ群に著しく偏っており，上位10アプリが全コネクション数の50%を占めていることである．特に，1位の「LINE」は単独で全コネクション数の約6分の1を占めており，携帯電話網を流れるトラヒックの性質に大きな影響を与えている．「LINE」は，主にテキストチャットを行うアプリであり，少量の通信を高頻度で行うため，制御信号数やユーザ端末の消費電力に影響を及ぼしやすい．2位の「Google連絡帳の同期」は，Android標準の連絡帳アプリのデータ同期を司るシステムアプリで，1位の「LINE」と合わせると，全コネクション数の25%を占める．ほかには，「Twitter」や「Facebook」といったSNSアプリ，「Chrome」や「Google Map」といった情報閲覧系アプリが，多くのユーザを獲得して上位にランクインしている．一方で，「Simeji」や「ウイルスバスター」といったアプリは，ユーザ数の少ない割にコネクション数が多い．したがって，トラヒックをユーザ単位で考える際には，これらのアプリの影響が相対的に大きくなる．

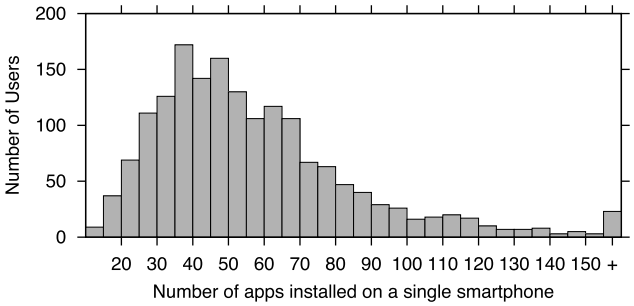

図1 スマートフォンにインストールされているアプリ数の分布
Fig. 1 Number of apps installed on a single smartphone.

表1 コネクション数の上位16アプリ
Table 1 Top 16 apps for connection count.

App Name	Connection#		User#
LINE	10,880,076	16.3%	1,116
Google Sync.	6,034,994	9.0%	1,471
Browser	4,317,009	6.4%	1,235
Facebook	3,333,681	5.0%	956
Twitter	2,740,469	4.1%	649
Google Search	2,149,407	3.2%	1,471
Simeji	1,334,526	2.0%	224
YouTube	1,220,967	1.8%	1,483
Google PLAY Store	1,180,899	1.8%	1,539
Chrome	1,017,815	1.5%	979
Yahoo!	1,012,369	1.5%	388
Google Map	848,033	1.3%	1,248
Google+	707,741	1.1%	1,087
Popacon (LINE)	599,103	0.9%	212
Messenger (Facebook)	525,953	0.8%	140
Virus Buster	500,287	0.7%	174

2.2.2 各アプリの利用状況

図2は，合計コネクション数の多い上位16アプリについて，通信間隔の累積分布関数を示したものである．通信間隔の中央値は，「ブラウザ」や「Twitter」などのアプリでは10秒前後であるが，「Google+」や「Gmail」などのアプリでは1,000秒を超えている．現在のLTE/3Gネットワークにおいて標準的なアイドルタイマ長である16秒以内での通信発生確率を見ても，10%〜60%とアプリによって大きく異なる．したがって，起動しているアプリの組合せによって，スマートフォンの生み出すトラヒックの性質は大きく変化し，通信制御の戦略にも影響を与える．

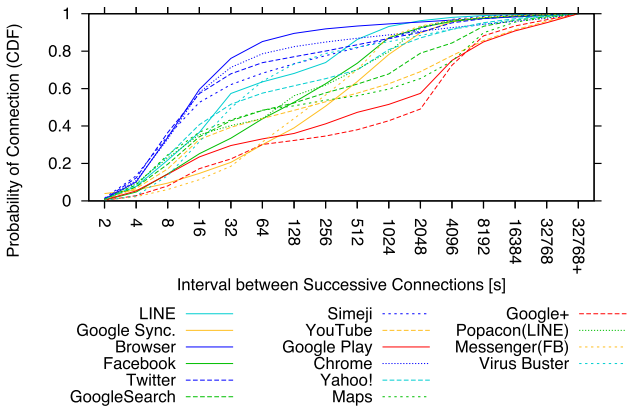

図2 上位16アプリの通信間隔の累積分布関数（CDF）
Fig. 2 Cumulative Distribution Function (CDF) of traffic intervals for top 16 apps.

2.2.3 画面の点灯状況との関係

前述の「通信量お知らせアプリ」では，各トラヒックの発生時に画面の点灯状況も記録している．**図 3** は，スマートフォンの稼働時間に対する画面点灯時間の割合の分布である．1,800 ユーザの平均値は 34.0%，中央値は 26.8%であり，一般的なスマートフォンは稼働時間の約 20%しか画面が点灯していない．一方，**図 4** は，各ユーザのトラヒックのうち画面点灯中に発生したものの割合の分布である．1,800 ユーザの平均値は 84.8%，中央値は 93.6%であり，一般的なスマートフォンではトラヒックの約 95%が画面点灯中に集中しているといえる．したがって，画面点灯中と画面消灯中では，トラヒックの発生確率に 100 倍近い差があることになる．したがって，画面の点灯状態を合わせて考慮することで，性能の向上を期待できる．

2.2.4 ユーザ操作との関係

一般に，ユーザがスマートフォンなどの情報通信機器を操作する場合，タッチパネルやボタンといった入力デバイスの操作タイミングとトラヒックの発生タイミングの間には，強い関連性があると考えられる．この仮説を検証するため，3 人の実験協力者を募り，合計 93 時間のトラヒック計測実験を行った．計測では，GALAXY Nexus SC-04D（Android 4.2.2）と NTT ドコモの 3G 回線を使用し，実験協力者にはなるべくふだんどおりにスマートフォンを利用してもらった．なお，この実験を前述の「通信量お知らせアプリ」に統合しなかったのは，Google PLAY を通して配布されるアプリの権限では，タッチパネルやボタンの操作状況を常時監視することが技術的に困難なためである．

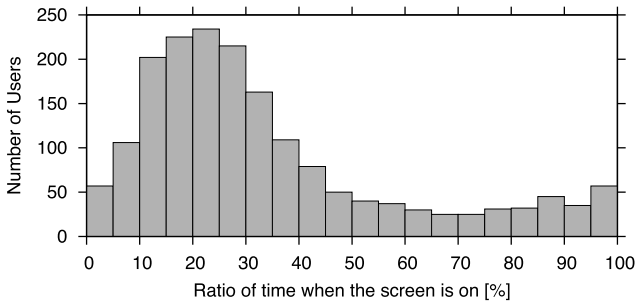

図 3 スマートフォンの稼働時間に対する画面点灯時間の割合のヒストグラム

Fig. 3 Histogram of ratio of time, when the screen is on, to the operating time of smartphones.

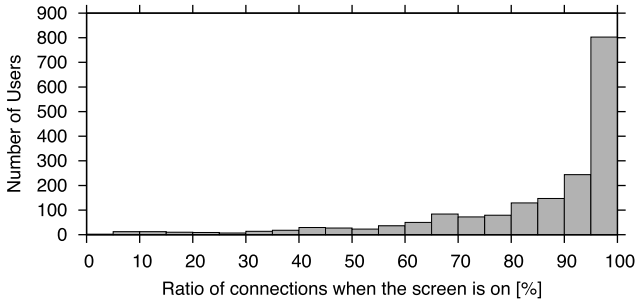

図 4 トラヒックのうち画面点灯中に発生したものの割合のヒストグラム

Fig. 4 Histogram of ratio of connections when the screen is on.

図 5 は，ユーザ操作からの経過時間を横軸に，累計のトラヒック発生確率を縦軸にとってプロットしたグラフである．あるユーザ操作があったとき，1 秒以内に通信が発生する確率は 15%，2 秒以内で 20%，5 秒以内で 26%となっており，ユーザ操作の有無が通信の発生確率を大きく左右する要因であることが確かめられた．したがって，ユーザ操作のタイミングを考慮することで，さらなる性能の向上を期待できる．

2.3 通信時の消費電力

スマートフォンのモバイルデータ通信による消費電力は，Radio Resource Control（RRC）における接続状態に応じて大きく変化する．RRC では，3G 端末の接続状態として *CELL_DCH*，*CELL_FACH*，*IDLE* の 3 状態が，LTE 端末の接続状態として *CONNECTED*，*IDLE* の 2 状態が定義されており，**図 6** に示すように通信状態に応じて状態遷移することになっている [12]．*IDLE* 状態への遷移は，無通信状態が一定時間続くことが条件となっており，この制御

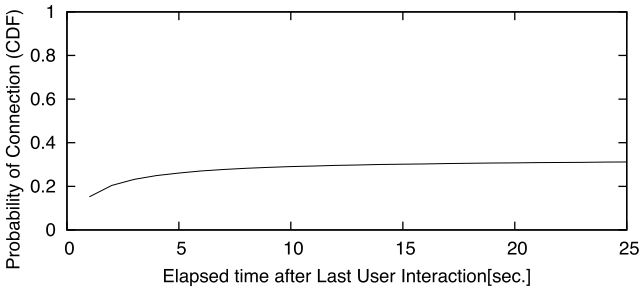

図 5 ユーザ操作と通信発生確率の関係．ユーザ操作から 5 秒以内に 26%の確率でトラヒックが発生する

Fig. 5 Relation between the user operation and probability of connection. 26% of traffic occurs within 5 seconds from user operations.

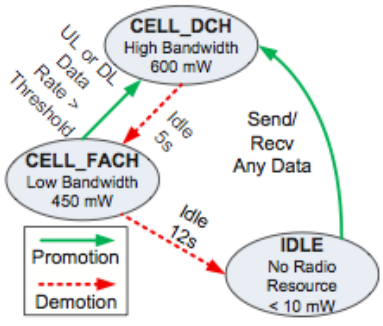

図 6 米国の携帯電話キャリアにおける RRC ステートマシンの一例．消費電力は Nexus One で計測したもの [13]

Fig. 6 RRC State Machine of a large UMTS carrier in the U.S.. The radio power consumption was measured on Nexus One. More details can be found in Ref. [13].

機構はアイドルタイマと呼ばれる．このタイマ時間の設定は，携帯電話事業者やユーザ端末の製造メーカに委ねられており，16 秒前後であることが多い．スマートフォンの通信時に電力計測を行うと，通信終了後に消費電力の大きい状態が尾を引くさまから，この 16 秒間を “Radio Tail” と呼ぶ．一般に Radio Tail を長くすると，コネクションを再利用できる確率が高まるため，コネクション制御にかかわる制御信号数を抑えられるものの，ユーザ端末の消費電力は増大する．反対に，Radio Tail を短くすると，ユーザ端末の消費電力は抑えられるものの，コネクションの切断と再確立が増加するため，制御信号数が増加しネットワークへの負荷が増大する．

2.2.2 項で述べた，アプリごとのトラヒックの性質の違いをふまえると，どのアプリが起動しているかによって，最適な通信制御の戦略が変わる．通信発生確率が高い状況では，アイドルタイマを延長して制御信号の削減を狙い，通信発生確率の低い状況では，アイドルタイマを短縮して消費電力の削減を狙う戦略が有効である．

我々は，実際の端末に設定されているタイマ時間と消費電力を確かめるため，電力計測実験を行った．この実験では，Android のリファレンス端末であり標準的な動作を期待できる GALAXY Nexus SC-04D（Android 4.1.1）と 0.1 Ω 抵抗，デジタルマルチメータ Agilent 34410A を用いて消費電力を計測した．**図 7** は，画面を消灯した状態で，30 秒ごとに数 KB のファイルを送受信したときの消費電力の推移である．図 7 では，消費電力は通信開始直後に 0.8 W まで急上昇し，約 5 秒後に 0.5 W に低下し，さらに約 8 秒後に元の 0.05 W まで急落する様子が読み取れる．これらの 3 状態はそれぞれ，RRC の *CELL_DCH*, *CELL_FACH*, *IDLE* に対応しており，この端末のアイドルタイマは合計 13 秒であることが分かる．本稿では，これらの値を用いて提案手法の制御信号および消費電力の削減性能を評価する．

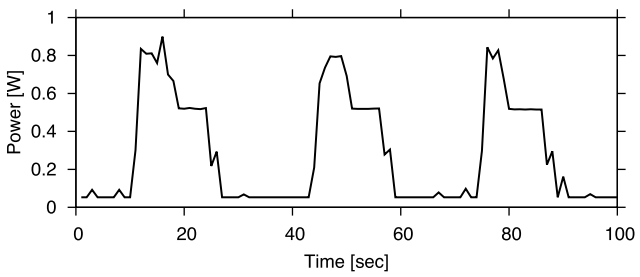

図 7 携帯電話網を通して通信を行った際の GALAXY Nexus（SC-04D）の消費電力

Fig. 7 Power consumption of GALAXY Nexus (SC-04D) during transmission on mobile network.

3. アプリ単位の通信制御

3.1 不要アプリの制御

冒頭で述べたように，スマートフォンにインストールされているアプリの中には，ユーザが必ずしもつねには必要としていないものが存在する．具体的には，Android OS に含まれている Google 系サービスのアプリ，メーカや通信事業者がプリインストールしたアプリ，ユーザ自身がインストールしたもののすでに使わなくなったアプリ，季節により利用頻度が大きく変化するアプリ，などが該当する．我々の実験では，未使用状態の「Google+」が約 1 時間周期で 100 KB 前後の定期通信を行うことが確認されている．

このような不要アプリのトラヒックを選択的に遮断することで，データトラヒックや制御信号，端末の消費電力を効果的に削減できる．ただし，誤ってユーザが利用中のアプリの通信を遮断してしまうとユーザへの影響が大きいため，本手法では以下の条件で随時，不要アプリの自動判定を行い，ユーザがアプリを利用し始めると自動的に制御対象から外れるようにする．利用状況の判定期間は，特定の曜日にだけ利用するアプリの存在を考慮して標準で 1 週間としたが，可変とすることでユーザが省電力と利便性のバランスを調整できるようになる．

- 当該アプリの起動回数が直近 1 週間で 0 回であること
- 通知バーへの通知回数が直近 1 週間で 0 回であること
- 当該アプリのウィジェットを設置していないこと
- ウェアラブル端末での利用がないこと
- Android OS のシステムアプリでないこと

ここでは，ユーザへのインタラクションの有無に焦点を当て，ユーザに何らかの情報提示をすることなく，バックグラウンドで通信するだけのアプリは不要と判定する．ただし，システム領域（`/system/app/`）にインストールされているアプリについては，OS の安定稼働のため制御対象外とする．これにより，3.3 節で述べるように，通信制御中であっても GCM によるプッシュ通知が利用可能となる．

しかしながら，上記の判定条件では，利用頻度の低いメーラやニュースリーダ，ウイルス対策アプリといったアプリが通信制御の対象となる可能性が残る．適切に実装されたアプリであれば，新着情報はプッシュ通知をトリガとして実行し，通信失敗時にはその旨の通知が表示されるため，大きな問題は発生しないが，そうでないアプリも数多く存在する．そこで，通信を遮断されたアプリの名前を通知領域にリアルタイムで表示し，そこから当該アプリを起動して一時的に通信制御の対象から外したり，通信制御の除外リストに追加したりできるようにする．また，直近 10 件ほどの遮断履歴も合わせて表示する．これにより，利用頻度の低いアプリには一時的な通信許可を，利用頻度の高いアプリには永続的な通信許可を，ユーザが手軽に与えることができる．

3.2 必要アプリの制御

不要アプリ以外のアプリは，通信を阻害するとユーザへの影響が大きいため，データトラヒックの制御は行わず，アイドルタイマを状況に応じて変化させることで，制御信号と消費電力のみを削減する．本手法では，アプリごとの通信間隔の分布と，画面の点灯状況，タッチパネルやボタンへのユーザ操作の有無に基づいて，アイドルタイマを動的に決定する．なお，これらの情報は Linux カーネル上で管理されており，dumpsys コマンドなどで取得可能である．

必要アプリに対する制御は LTE/3G 接続中のみを対象とし，コネクション管理に制御信号を必要としない Wi-Fi 接続中は制御を行わない．また，Wi-Fi 接続中はアプリの挙動が変化する場合があるため，後述する各アプリのトラヒックの性質調査も停止する．ユーザによっては，Wi-Fi 接続と LTE/3G 接続を頻繁に行き来することがあるが，トラヒックの性質調査に時間がかかることを除けば，性能面への大きな影響はない．将来的には，Wi-Fi 接続中のアプリの挙動変化や Wi-Fi モジュールの消費電力も考慮し，LTE/3G 接続と Wi-Fi 接続の自動切替え制御を盛り込んで，制御信号と消費電力のさらなる最適化を検討する．

3.2.1 アプリごとの通信間隔の分布に基づく制御

本手法では，まず，事前に収集したアプリごとの通信間隔の分布データ（図 2）と実行中のアプリの一覧を用いて，その時点での端末全体のトラヒックの性質を予測する．実行中のアプリ一覧は Android SDK の ActivityManager API で取得し，この一覧に含まれる各アプリの通信間隔の累積分布関数（$CDF_{App\#1}$，$CDF_{App\#2}$,$\cdots$,$CDF_{App\#N}$）から，以下の計算式を用いて端末全体のトラヒックの通信間隔の累積分布関数 CDF_{Total} を算出する．

$$CDF_{Total}(t) = 1 - \prod_{n=1}^{N}\{1 - CDF_{App\#n}(t)\} \qquad (1)$$

ここで，t は直近の通信からの経過時間である．なお，各アプリの通信頻度の情報は，分布データに含まれるため，ここで重み付けは必要ない．

次に，得られた端末全体の通信間隔の分布データから，最適なアイドルタイマ長を決定する．アイドルタイマの調整では，制御信号数と消費電力がトレードオフの関係となるため，なるべく短いアイドルタイマで消費電力を抑えつつ，Radio Tail 部分でなるべく多くのトラヒックをカバーして制御信号を削減することが望ましい．ここで，単位消費電力あたりの制御信号削減量を「アイドルタイマ効率」と定義すると，それはすなわち，単位アイドルタイマ長あたりの後続トラヒックカバー率であり，カバー率が 100%となる点において，許容されるアイドルタイマ長の上限が定まる．また，通信間隔の CDF のグラフ（**図 8**）では，時間軸を線形軸とした場合の原点からの傾きにあたる．この傾きが基準値より大きくなる区間のうち，アイドルタイマが最長となる点を選ぶことで，所望のアイドルタイマ効率を維持しつつ，トラヒックに応じたアイドルタイマ長が得られる．ただし，通信の発生頻度が低い場合には，この条件を満たす点が存在しないため，3G ネットワークの RTT（数百 ms）とコンテンツサーバの応答時間を考慮し，1 回の通信が分断されることのないよう，アイドルタイマ長は 2 秒とする．

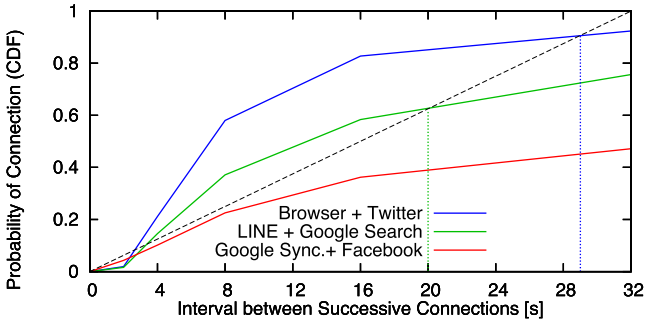

図 8 通信間隔の CDF において，原点からの傾きがアイドルタイマ効率を意味する．図 2 と違い，時間軸は線形である

Fig. 8 In CDF of communication intervals, the slope from the origin means the idle timer efficiency. Note that time-axis is linear unlike Fig. 2.

図 8 は，アイドルタイマ長の上限を 32 秒と設定した場合，すなわち，アイドルタイマ効率の基準値を 100%/32 sec = 3.125(%/sec) と設定した場合の例である．この基準値では，「ブラウザ」と「Twitter」の合成トラヒック（青線）に最適なアイドルタイマは 29 秒となり，「LINE」「Google 検索」の合成トラヒック（緑線）では 20 秒となる．ただし，「Google 連絡先の同期」と「Facebook」の合成トラヒック（赤線）では，アイドルタイマ効率が 3.125(%/sec) 以上となる点が存在しないため，アイドルタイマは 2 秒とする．なお，このアイドルタイマ効率の基準値は，制御信号数と消費電力のどちらを優先したいかに応じて適宜変更できる．

3.2.2 ユーザ操作に基づく制御

ユーザ操作の影響については，図 5 に示した通信発生確率の $CDF_{operation}$ を，以下の式を用いて，全アプリの合成トラヒックの CDF に反映する．すなわち，ユーザ操作から τ 秒経過後に発生するトラヒックの割合を乗算することで，ユーザ操作の影響を反映する．

$$CDF(t) = CDF_{Total}(t)\{1 - CDF_{operation}(\tau)\} \qquad (2)$$

ここで，t は直近の通信からの経過時間，τ は直近のユーザ操作からの経過時間である．t と τ は必ずしも一致しないことに注意が必要である．

3.2.3 画面の点灯状況に基づく制御

2.2.3 項で確認したように，画面の点灯状況によって通信の発生頻度は大きく変化する．これに対応するため，3.2.1 項の制御では，画面点灯中と消灯中に分けて通信間隔の分布を集計し，画面の点灯状況に応じた分布データを

用いて，端末全体のトラヒックの通信間隔の累積分布関数 CDF_{Total} を算出し，最適なアイドルタイマ長を決定する．

以上の手順で算出したアイドルタイマ長は，以下のタイミングで再計算し適用する．

- 端末上のいずれかのアプリが起動または終了したとき
- 端末上のいずれかのアプリが通信を行ったとき
- 端末の画面が点灯または消灯したとき
- 端末の電源が投入されたとき

3.3 不要アプリの制御機構

3.3.1 概要

本手法では，Android SDK でサポートされていない，アプリ単位での通信制御を実現するため，Linux カーネルの `iptables` コマンドを活用する．

`iptables` のパケットフィルタには，INPUT/FORWARD/OUTPUT の 3 つのルールチェインがあり，それぞれ，その端末に到着/通過/送出されるパケットに対する処理を管理している．各ルールでは，送信元 IP アドレスや宛先ポート番号などを指定して処理対象とするパケットを規定し，通過/拒否/破棄などの処理を指定できる．本手法では，OUTPUT チェインのみを使用する．これは，INPUT チェインで受信パケットを遮断したとしても，そのパケットはすでに携帯電話ネットワークを通して配送されたものであり，コネクションの確立を阻止できないためである．なお，OUTPUT チェインでは，送信元プロセスの UID を指定して，処理対象とするパケットを規定できる．

一方，Android OS では，アプリは各々に割り当てられた Dalvik 仮想マシン上で実行される仕組みになっている．それぞれの Dalvik 仮想マシンは，アプリに割り当てられた固有の UID で実行されており，ファイル IO やネットワーク通信などカーネルレベルの命令を実行する場合には，その UID と Linux カーネルのユーザ管理機能を用いて，アプリ間の独立性を確保している．それゆえ，どのアプリがどのファイルにアクセスしたか，あるいは，どのアプリがどれだけネットワーク通信を行ったか，を Android OS では簡単に調べることができる．

そこで，本手法では，`iptables` の UID 指定機能を用いて，アプリ単位での通信制御を実現する．なお，`iptables` の利用には root 権限が必要となるため，本手法は OS ベンダまたは端末メーカによる実装を想定している．

3.3.2 プッシュ通知のサポート

本研究では，SMS や GCM（Google Cloud Messaging）によるプッシュ通知がユーザビリティの維持に不可欠であると考え，通信制御中であってもプッシュ通知が利用可能となるよう配慮している．制御プレーンを通して配送される SMS は，我々の通信制御による影響をまったく受けないため，ここでは GCM によるプッシュ通知を利用可能とするために必要な要件について論じる．

まず，GCM の仕組みについて簡単に説明する．端末上で GCM のプッシュ通知を利用するアプリが起動されると，Android OS は GCM サーバにアクセスしてユーザトークンを登録し，GCM サーバとの間に TCP コネクションを確立する．その後は，定期的にパケットをやりとりすることでコネクションを維持し，何らかの理由でコネクションが切断された場合には自動的にコネクションを張り直す．具体的には，Wi-Fi 接続中は 15 分に 1 度，LTE/3G 接続中は 53 分に 1 度，`mtalk.google.com` の 5,228〜5,230 番ポートと通信することが知られている．

次に，GCM サーバにメッセージの配信依頼が届くと，宛先のユーザトークンから端末を特定し，その端末との間のコネクションを調べる．もし，コネクションが存在すれば，そのコネクションを通じて端末にメッセージを送信する．もし，コネクションが切断されていた場合には，端末がオフライン状態にあると判断し，次にコネクションが確立されたときにメッセージを送信する．

端末側では，サーバからメッセージが届くと，Android OS が GCM サーバに ACK を返し，Broadcast Intent を用いて当該アプリにメッセージが伝えられる．GCM の通信部分はすべて Android OS によって管理されており，アプリはユーザトークン登録の API を 1 度コールすれば，後は Broadcast Intent を待つだけでよい仕組みになっている．それゆえ，本提案手法のように，特定のアプリの UID を指定して通信を遮断した場合でも，Android OS のシステムを司る UID を制御対象としない限りは，当該アプリは正常にプッシュ通知を受信することができる．ただし，プッシュ通知をトリガとして，当該アプリがメールの取得やデータの同期などの通信を行う場合には，通信制御の対象となってしまうため，別途配慮が必要である．

一方，`root`，`system`，`shell` など，Android OS のシステムを司る UID を指定して通信を遮断した場合や，UID を指定せずに全通信を遮断した場合には，GCM サーバと端末の間のコネクションを維持できなくなるため，プッシュ通知の配信は不可能となる．この場合，GCM サーバは端末がオフライン状態になったと判断し，次回のコネクション確立までメッセージの送信を待機する．

本提案手法では，GCM への影響を防ぐため，システム領域（`/system/app/`）のアプリを制御対象から除外し，ユーザ領域（`/data/app/`）のアプリのみを制御対象とする．

3.3.3 他の実装方式の検討

Android 端末上で通信制御を行う手法は，`iptables` 以外にも存在する．root 権限を必要とせず，Android SDK の範囲内でアプリごとの通信制御を実現する手段としては，VpnService API があげられる．しかし，この方式では制御用の Android アプリが常駐するため，Linux カーネル上で動作する `iptables` 方式と比較して端末の CPU 負荷が大きい．一方，root 権限が利用可能であれば，端末上にプ

ロキシサーバを設置する方式や，hosts ファイルを編集する方式が考えられる．しかし，前者は，アプリごとの制御を実現するためにパケット解析が必要で CPU 負荷が大きく．後者は，ホスト名での制御しかできない．ほかに，外部サーバを経由させる方式も考えられるが，ユーザ数に比例してサーバの負荷が増大し，サービスとしてスケールしない．

3.4 必要アプリの制御機構

アイドルタイマ長の制御方法に関しては，各端末のチップセットや実装に依存するため，汎用的な実装方法に関する言及は避ける．ここでは一例として，ソニーモバイルの Xperia シリーズにおけるアイドルタイマの設定方法を紹介する．

Xperia シリーズでは，省電力化のためにアイドルタイマ長を短縮する実装がなされており，そのコンポーネントが Linux カーネル上に散見される．Xperia Z1 f（SO-02F）の場合，RRC コネクション管理のためのコマンド（`/system/bin/fast-dormancy`）と設定ファイル（`/system/etc/fast-dormancy/fd_int_conf.txt`）が存在する．設定ファイルには waitTime という項目があり，初期値は “5” 秒となっている．この項目を編集することで，アイドルタイマ長を変更できる．なお，Samsung，LG 電子，HTC，モトローラ，シャープ，富士通製の端末についても調査を行ったが，root 権限を取得できない端末も多く，同様のコンポーネントを発見することができなかった．

なお，上記コマンドの実行や，上記ファイルの編集には root 権限が必要となるため，3.3 節で述べた不要アプリに対する制御と同じく，OS ベンダまたは端末メーカによる実装を想定している．

4. 制御信号と消費電力の削減量の評価

4.1 不要アプリに対する性能

4.1.1 評価方法

我々は，前述の「通信量お知らせアプリ」を用いて，通信発生時の各アプリの実行状況を統計的に調査した．具体的には，Android SDK の ActivityManager.RunningAppProcessInfo API を用い，通信発生時に 10 回に 1 回の割合で，各アプリによるユーザへのインタラクションの有無を以下の 6 段階で取得した．前半の 3 段階はユーザにとって認知可能な状態，後半の 3 段階は認知不能な状態である．

- 100：フォアグラウンドで動作中
- 130：BGM 再生などユーザが認知可能な状態
- 200：ウィジェットなどが可視な状態
- 300：サービスとしてバックグラウンドで動作中
- 400：何らかのコードがバックグラウンドで動作中
- 500：休止中

ここでは，直近 1 カ月間のデータにおいて少なくとも 1 回以上，ユーザにとって認知可能な状態になったことのあるアプリを必要アプリとし，それ以外を不要アプリとして，コネクション数の削減量を評価した．なお，ユーザ数が少ないのは，この機能を追加した時期が遅かったためである．

4.1.2 コネクション数の削減効果

調査対象となった 419 ユーザの全コネクションのうち，13.1%が不要アプリによるものであった．したがって，必要アプリのトラヒックとの兼ね合いのため一概にはいえないが，不要アプリのトラヒック遮断により最大で 13.1%の制御信号と消費電力を削減できる．また，ユーザ別に見ると，不要アプリのコネクションが全体の 38.5%に達するユーザもおり，削減効果はユーザの利用状況によって大きく異なる．

4.2 必要アプリに対する性能

4.2.1 評価方法

我々は，2 章で収集したトラヒックの統計データを用いて仮想的なトラヒックを生成し，提案手法を適用した際の制御信号と消費電力の削減量を評価した．ただし，ユーザ操作のタイミングについては，「通信量お知らせアプリ」で収集したデータには含まれないため，4.2.4 項で別途評価を行うこととし，ここではユーザ操作のない状況を想定した．また，制御信号数はコネクション数に依存して決まり通信データ量には依存しないこと，データの送受信に必要な電力は必要経費であることから，ここではデータ量は考慮せず Radio Tail 部分の消費電力に焦点を当てている．また，ディスプレイや CPU の消費電力は考慮していない．

まず，表 1 に示したコネクション数での上位 16 アプリから，1〜16 個を無作為に選定し，3.2 節の式 (1) を用いて，合成トラヒックの通信間隔の累積分布関数を得た．次に，この分布に基づいて，10,000 コネクションの仮想的なトラヒック間隔を生成した．この仮想トラヒックに対し，2 章で確認した実際の消費電力（Radio Tail 部分：0.517 W，アイドル時：0.056 W）とアイドルタイマ長（13 秒）を用いて，本手法の動的アイドルタイマ適用時と，従来の固定長アイドルタイマ適用時の制御信号数と消費電力をそれぞれシミュレーション評価した．シミュレーションは，画面点灯中・消灯中の各データに対して，アプリ数を 1〜16 個に変化させながら，各 100 回ずつ行った．なお，**図 9**，**図 10** の縦軸は，従来方式を 100%とする相対値であり，エラーバーは標準偏差を表す．

4.2.2 実行中のアプリ数と削減性能

図 9 は，許容するアイドルタイマ長の上限を 32 秒と設定した場合における，画面点灯時の制御信号および消費電力の削減量である．各シミュレーションで選択されたアプリの組合せによって，性能に大きなバラつきがあるものの，制御信号を大幅に削減できる．実行中のアプリ数が多くト

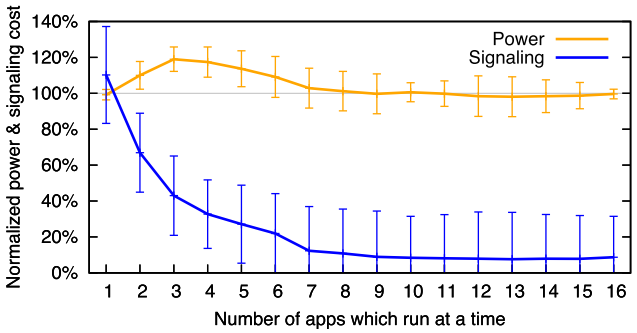

図 9 従来方式を 100%とした，画面点灯時の提案手法の性能評価
Fig. 9 Evaluation of the proposal method when screen is on, compared to static idle timer as 100%.

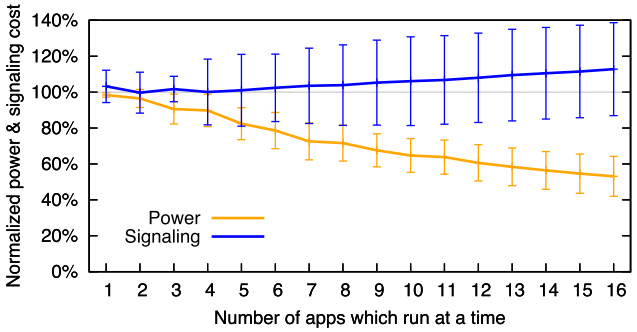

図 10 従来方式を 100%とした，画面消灯時の提案手法の性能評価
Fig. 10 Evaluation of the proposal method when screen is off, compared to static idle timer as 100%.

ラヒックの発生頻度が高いほど削減性能は向上し，消費電力への影響もアプリ数 3 をピークに減少する．アプリが 16 個の場合には，制御信号を平均 91.3%削減しつつ，消費電力を平均 0.4%削減できる．

一方，画面消灯時の制御信号および消費電力の削減量は図 10 のとおりである．画面点灯中と比較すると削減幅は小さいものの，消費電力を削減できる．実行中のアプリ数が多いほど性能が向上し，アプリが 16 個の場合には，消費電力を平均 46.9%削減しつつ，制御信号の増加を平均 12.8%に抑えられる．

2.2.3 項で指摘したように，スマートフォンは稼働時間の約 20%の時間しか画面が点灯した状態にないが，トラヒックの 95%は画面点灯中に発生する．このことをふまえて，点灯中と消灯中を合わせた全体での性能を評価する．消費電力は時間に依存するため，全体では $-0.4 \times 0.20 - 46.9 \times 0.80 = -37.6\%$ の消費電力を削減できることになる．一方，制御信号数はトラヒック量に依存するため，全体では $-91.3 \times 0.95 + 12.8 \times 0.05 = -86.1\%$ の制御信号を削減できることになる．

4.2.3 アイドルタイマ効率の基準値と削減性能

図 11 は，実行中のアプリ数が 10 個の場合において，アイドルタイマ効率の基準値，すなわち，許容されるアイドルタイマ長を変化させたときの制御信号および消費電力の削減量である．当然ながら，許容するアイドルタイマ長を長くするほど，制御信号の削減幅は大きくなるが，消費電力の削減幅は次第に小さくなっていく．反対に，許容するアイドルタイマ長を短くすると，制御信号が急激に増加する．制御信号の削減幅は 32 秒付近から伸びが鈍くなっているため，アイドルタイマ効率の基準値を 100%/32 sec = 3.125(%/sec) とするのが良い．ただし，この基準値は，制御信号数と消費電力のどちらを優先したいかに応じて適宜変更できる．

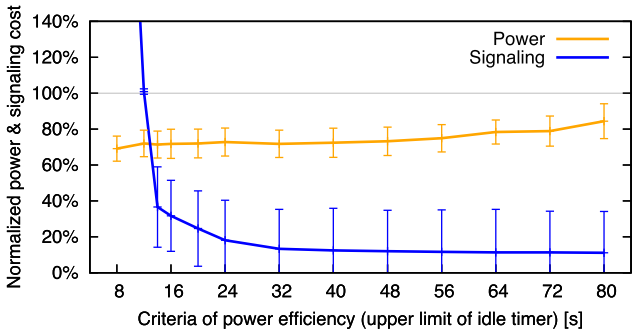

図 11 アイドルタイマ効率の基準値を変化させたときの性能評価
Fig. 11 Evaluation of the proposal method for each idle timer efficiency.

4.2.4 ユーザ操作の考慮と削減性能

我々は，2.2.4 項で収集した，ユーザ操作およびトラヒックの発生間隔に関する 3 ユーザ分の統計データを用いてシミュレーション評価を行った．なお，評価を独立させたのは，「通信量お知らせアプリ」でユーザ操作に関する情報を収集できなかった関係で，シミュレーションに用いる統計データが異なるためである．

まず我々は，統計データに基づいて 86,400 秒分の仮想的なタッチ操作およびトラヒック発生の時系列を生成した．なお，この統計データでは，アプリごとの通信は区別されておらず，端末全体のトラヒックを一括りにしている．次に，この仮想トラヒックに対し，2 章で確認した実際の消費電力（Radio Tail 部分：0.517 W，アイドル時：0.056 W）とアイドルタイマ長（13 秒）を用いて，制御信号と消費電力の削減量を評価した．シミュレーションは，アイドルタイマ効率の基準値を変化させながら，各 10 回ずつ行い，グラフには平均と標準偏差を示す．4.2.1 項と同様，通信データ量は考慮せず，Radio Tail 部分の消費電力に焦点を当てている．

図 12 は，アイドルタイマ効率の基準値を変化させたときの制御信号および消費電力の削減量である．設定された基準値に応じて，消費電力または制御信号数を削減できる．4.2.3 項で述べたように許容するアイドルタイマ長を 32 秒とした場合には，平均 0.7%の制御信号増で平均 2.5%の消費電力を削減可能である．4.2.3 項までの評価結果とは，シミュレーションのもととなった統計データが異なるため単

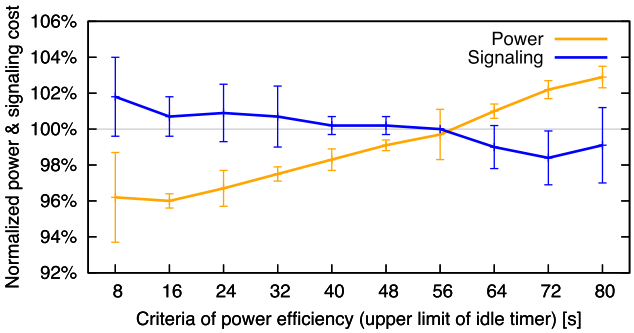

図 12 ユーザ操作のみを考慮した場合の提案手法の性能評価
Fig. 12 Evaluation of the proposal method only with consideration for user operation.

純な掛け算はできないが，ユーザ操作を考慮することで，消費電力と制御信号をより良く削減できると考えられる．

5. 関連研究

スマートフォンの通信制御と省電力化に関しては，様々な研究調査がさかんに行われてきた [14], [15], [16], [17]．しかし，我々の知る限りでは，1 つ 1 つのアプリについて個別にトラヒックの性質を分析し，通信間隔の分布に着目して制御に反映させることを検討する研究はなかった．Chakraborty らは，スマートフォンのバックグラウンド通信が，同じ基地局に接続している他の端末の通信と重ならないよう，各端末が分散的にスケジューリングを行うことで，端末の消費電力を削減しつつ平均スループットを向上させる画期的な手法を提案している [18]．Athivarapu らは，スマートフォン上のアプリが発行するシステムコールに着目し，その発行パターンから通信タイミングを予測し，ごくわずかの制御信号増と引き換えに，ユーザ端末の消費電力を 20〜40%削減する手法を提案した [19]．この手法は消費電力の削減に重きをおいており，制御信号数の削減については詳しく考慮されていない．Huang らは，スマートフォンの画面消灯中に行われる通信の多くが，画面点灯中の通信と比べて，高い遅延耐性を持つことを発見し，スリープ中の通信をスケジューリングすることで，消費電力を約 60%削減する手法を提案した [20]．この研究では，20 ユーザ・5 カ月分の通信記録しか解析しておらず，トラヒックの内容に偏りがあった可能性がある．Qian らは，スマートフォン上のアプリが，自身の行う通信の開始タイミングや終了タイミングを正確に予測できることに着目し，アプリが OS に対して通信タイミングを通知する仕組みを構築することで，Radio Tail の長さを最適化する手法を提案した [21]．しかしながら，この手法ではアプリ側での対応が必要となり，100 万本を超える既存のアプリ資産をそのまま活用することができないという欠点がある．特定の種類の通信に特化した手法としては，Han らによるテザリング通信に特化した省電力化手法 [22] や，Xu らによるメール同期タイミングの最適化手法 [23]，Nath らによるモバイルアプリ向け広告の通信最適化手法 [24] などがあげられる．これらの手法は，いずれも我々の提案手法と組み合わせることが可能であり，さらなる性能向上を期待できる．

6. まとめ

本稿では，スマートフォン上の様々なアプリに対して，各アプリの性質を反映した通信制御を個別に実施することで，ユーザビリティを維持しつつ，携帯電話網を流れるデータトラヒックと制御信号，端末の消費電力を効果的に削減する手法を提案した．

本手法では，アプリ起動やウィジェット利用の有無などの情報に基づいて不要アプリを判別し，それらの通信を選択的に遮断することで，効果的にトラヒックを削減する．また，他のアプリについては，各アプリのトラヒックの統計的な性質に基づきアイドルタイマを動的に変更することで，制御信号量と消費電力を削減する．また，画面の点灯状態やユーザ操作からの経過時間も考慮する．

本手法では，アプリ単位での通信制御が鍵となるが，Android SDK にはそれを実現する API が存在しないため，我々は，Android が Linux カーネル上で動作するミドルウェアであることに着目し，iptables コマンドを用いて制御機構部分を実装した．

評価実験では，実際に稼働中の Android 端末から収集したトラヒックの統計データを用いて仮想トラヒックを生成し，本提案手法を適用した場合のトラヒックをシミュレーションして，制御信号量と消費電力の削減量を評価した．結果として，制御信号を最大 86.1%，消費電力を最大 37.6%，同時に削減できることを確認した．

参考文献

[1] 高木 雅，川原圭博，浅見 徹：3G トラヒック削減とプッシュ通知を共存させるアプリ単位の通信制御手法，DICOMO2013 シンポジウム，7F-2 (2013).
[2] 高木 雅, 川原圭博, 浅見 徹：スマートフォン・トラヒックのアプリ単位での分析，信学総大，No.B-15-7 (2014).
[3] Ericsson: Ericsson Mobility Report, Ericsson (online), available from ⟨http://www.ericsson.com/res/docs/2013/emr-august-2013.pdf⟩ (accessed 2014-04-27).
[4] Ericsson: Traffic and Market Report, Ericsson (online), available from ⟨http://www.ericsson.com/res/site_JP/press/2012/doc/Traffic_rep_and_smartphone_issues_120619.pdf⟩ (accessed 2014-04-27).
[5] NTT ドコモ：2012 年 1 月 25 日の FOMA 音声・パケット通信サービスがご利用しづらい状況について，NTT ドコモ（オンライン），入手先 ⟨http://www.nttdocomo.co.jp/binary/pdf/info/news_release/2012/01/26_00.pdf⟩（参照 2013-05-11）.
[6] Carroll, A. and Heiser, G.: An Analysis of Power Consumption in a Smartphone, *USENIX ATC*, pp.21–21 (2010).
[7] Falaki, H., Mahajan, R., Kandula, S., Lymberopoulos, D., Govindan, R. and Estrin, D.: Diversity in Smart-

phone Usage, *ACM MobiSys*, pp.179–194 (2010).
[8] Carolan, E., McLoone, S.C. and Farrell, R.: Comparing and Contrasting Smartphone and Non-Smartphone Usage, *WWW ISSC* (2013).
[9] Qian, F., Wang, Z., Gao, Y., Huang, J., Gerber, A., Mao, Z., Sen, S. and Spatscheck, O.: Periodic transfers in mobile applications: Network-wide origin, impact, and optimization, *ACM WWW*, pp.51–60 (2012).
[10] Kang, J.-M., Seo, S.-S. and Hong, J.W.-K.: Usage pattern analysis of smartphones, *IEEE APNOMS*, pp.1–8 (2011).
[11] 高木 雅：通信量お知らせアプリ，Google PLAY（オンライン），入手先 ⟨https://play.google.com/store/apps/details?id=akg.traffic.notifier⟩（参照 2014-04-27）.
[12] GSMA: Fast Dormancy Best Practices, GSMA (online), available from ⟨http://www.gsma.com/newsroom/ts18-v10-tsg-prd-fast-dormancy-best-practices/⟩ (accessed 2014-06-30).
[13] Qian, F., Wang, Z., Gerber, A., Mao, Z.M., Sen, S. and Spatscheck, O.: Characterizing radio resource allocation for 3G networks, *ACM IMC*, pp.137–150 (2010).
[14] Pathak, A., Jindal, A., Hu, Y.C. and Midkiff, S.P.: What is keeping my phone awake?: Characterizing and detecting no-sleep energy bugs in smartphone apps, *ACM MobiSys*, pp.267–280 (2012).
[15] Schulman, A., Navda, V., Ramjee, R., Spring, N., Deshpande, P., Grunewald, C., Jain, K. and Padmanabhan, V.N.: Bartendr: A practical approach to energy-aware cellular data scheduling, *ACM MobiCom* (2010).
[16] Zhao, B., Zheng, Q., Cao, G. and Addepalli, S.: Energy-Aware Web Browsing in 3G Based Smartphones, *IEEE ICDCS*, pp.165–175 (2013).
[17] 山本享弘，猿渡俊介，南 正輝，森川博之：Piggyback Transport Protocol：Participatory Sensing における低消費電力なアップロードエンジン，情報処理学会論文誌，Vol.53, No.1, pp.274–285 (2012).
[18] Chakraborty, A., Navda, V., Padmanabhan, V.N. and Ramjee, R.: Coordinating cellular background transfers using loadsense, *ACM MobiCom*, pp.63–74 (2013).
[19] Athivarapu, P.K., Bhagwan, R., Guha, S., Navda, V., Ramjee, R., Arora, D., Padmanabhan, V.N. and Varghese, G.: RadioJockey: Mining program execution to optimize cellular radio usage, *ACM MobiCom*, pp.101–112 (2012).
[20] Huang, J., Qian, F., Mao, Z.M., Sen, S. and Spatscheck, O.: Screen-off traffic characterization and optimization in 3G/4G networks, *ACM IMC*, pp.357–364 (2012).
[21] Qian, F., Wang, Z., Gerber, A., Mao, Z.M., Sen, S. and Spatscheck, O.: TOP: Tail Optimization Protocol For Cellular Radio Resource Allocation, *IEEE ICNP*, pp.285–294 (2010).
[22] Han, H., Liu, Y., Shen, G., Zhang, Y. and Li, Q.: DozyAP: Ppower-efficient Wi-Fi tethering, *ACM MobiSys*, pp.421–434 (2012).
[23] Xu, F., Liu, Y., Moscibroda, T., Chandra, R., Jin, L., Zhang, Y. and Li, Q.: Optimizing background email sync on smartphones, *ACM MobiSys*, pp.55–68 (2013).
[24] Nath, S., Lin, F.X., Ravindranath, L. and Padhye, J.: SmartAds: Bringing contextual ads to mobile apps, *ACM MobiSys*, pp.111–124 (2013).

推薦文

スマートフォンにインストールされている様々なアプリに対して，アプリ単位で通信制御を行うことで，スマートフォンの 3G トラヒックと消費電力を効果的に削減する手法を提案しており，新規性も高く検証も十分されており非常に有用な内容であり，論文誌に推薦する価値がある．

（コンシューマ・デバイス&システム研究会主査 石川憲洋）

高木 雅 （学生会員）

1990 年生．2013 年東京大学工学部電子情報工学科卒業．同大学大学院修士課程に在学中．モバイルネットワークの最適化に関する研究に従事．電子情報通信学会会員．

川原 圭博 （正会員）

1977 年生．2000 年東京大学工学部電子情報工学科卒業．2002 年同大学大学院修士課程修了，2005 年同博士課程修了．博士（情報理工学）．2005 年同大学院情報理工学系研究助手．助教を経て，2010 年同講師，2013 年同准教授．2011～2013 年ジョージア工科大学客員研究員および MIT Media Lab 客員教員を兼任．2014 年 AgIC 株式会社技術アドバイザー．電子情報通信学会，IEEE，ACM 各会員．

浅見 徹 （正会員）

1952 年生．1974 年京都大学工学部電子情報工学科卒業．1976 年同大学大学院修士課程修了．同年国際電信電話株式会社（KDDI）に入社．UNIX 通信，ネットワーク障害診断，xDSL の実証実験等に従事．博士（情報理工学）．2001 年 KDDI 研究所代表取締役所長．2006 年東京大学教授．電子情報通信学会，IEEE，ACM 各会員．