情報処理学会論文誌 Vol.53 No.7 1863–1874 (July 2012)

推薦論文

Lightweight Lower-than-Best-Effort：携帯電話センシングのための軽量な低優先度通信機構

山本 享弘[1,2,a)] 猿渡 俊介[1] 森川 博之[1]

受付日 2011年11月9日, 採録日 2012年4月2日

概要：スマートフォンに代表される近年の携帯電話の高機能化にともない，携帯電話に搭載されたセンサを使って実空間情報を収集する携帯電話センシングが実現可能となった．しかしながら，バックグラウンドで動作する携帯電話センシングの処理や通信が，フォアグラウンドのウェブブラウジングなどのネットワークアプリケーションの通信に遅延を生じさせるという問題が発生する．本稿では，携帯電話センシングを実現したときのユーザ通信の遅延を抑制するために，携帯電話上でトラヒックの優先制御を行うLW-LBE（Lightweight Lower-than-Best-Effort）の設計と実装，評価について述べる．LW-LBEでは，実行オーバヘッドの小さいソフトウェア割込みハンドラをプライオリティキュー単位に割り当て，タスクとソフトウェア割込みハンドラの間で実行順序を制御することで，低優先度通信を低オーバヘッドで実現する．実機上にLW-LBEを実装して評価を行った結果，既存手法よりも少ない実行サイクル数でフォアグラウンド通信の遅延が削減できることを示す．

キーワード：携帯電話，優先制御，優先度逆転，ユーザ参加型センシング，センサネットワーク

Lightweight Lower-than-Best-Effort for Mobile Phone Sensing

Takahiro Yamamoto[1,2,a)] Shunsuke Saruwatari[1] Hiroyuki Morikawa[1]

Received: November 9, 2011, Accepted: April 2, 2012

Abstract: Mobile phone sensing collects real-world information with sensors on mobile phones. The sensing applications get the real world information and upload sensor data to the server periodically in background. The sensor data uploading induces communication delay of user's network applications such as web browsing. To reduce the communication delay, this paper shows Lightweight Lower-than-Best-Effort (LW-LBE) protocol, which sets the lower priority than best-effort to the background traffic. The LW-LBE assigns software-interrupt handlers to each priority, and controls the execution order among software-interrupt handlers and tasks. We implemented the LW-LBE on an android phone and evaluated the LW-LBE. The evaluation results show LW-LBE reduces the communication latency of the foreground application to the same level with the previous work and reduces more CPU load than the previous work.

Keywords: mobile phone, priority control, priority inversion, participatory sensing, sensor network

1. はじめに

携帯電話の高度化によって登場したスマートフォンが具備するセンサを用いることで，時間的，空間的にもこれまでにない粒度・量の実空間情報を収集することが可能になりつつある [1]．スマートフォンを利用した携帯電話センシングでは，ユーザの操作をともなわずにバックグラウンドで定期的に通信を行うことも想定されている．この結果，フォアグラウンドでユーザがウェブの閲覧を行っている間

1 東京大学先端科学技術研究センター
Research Center for Advanced Science and Technology, the University of Tokyo, Meguro, Tokyo 153–8904, Japan
2 株式会社コア先端組込み開発センター
CORE Advanced Embedded Technology Development Center, Kawasaki, Kanagawa 215–0034, Japan
a) tak-yama@mlab.t.u-tokyo.ac.jp

本稿の内容は2011年6月のモバイルコンピューティングとユビキタス通信研究会にて報告され，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．

にバックグラウンドでセンシングデータのアップロードが行われると，ウェブ閲覧のためのスループットが低下してユーザによるフォアグラウンドの操作や通信が遅延するという問題が生じる．

バックグラウンドのアプリケーションがフォアグラウンドに影響を与えないようにするためには，タスク処理と通信処理の双方から考える必要がある．たとえば，ユーザが操作している間にバックグラウンドのアプリケーションに処理が遷移したり，ユーザの通信が発生している間にバックグラウンドの通信が発生したりすることは避けなければならない．特に携帯電話では計算資源や通信資源が PC に比べて少ないため，タスクとパケットの 2 つの優先制御を同時かつ低オーバヘッドに行う仕組みが求められる．

このような観点から，本稿では，携帯電話において低オーバヘッドで実現できるパケットとタスクの低優先度スケジューリング機構「Lightweight Lower-than-Best-Effort」（LW-LBE）を示す．LW-LBE では，フォアグラウンドの通信が発生してからトラヒックの制御が開始されるまでの遅延を削減するために，IP 層でトラヒックの優先制御を行う．また，低オーバヘッドでトラヒックの優先制御を実現するために，プライオリティキューに格納されたパケットの処理をオーバヘッドの小さいソフトウェア割込みハンドラによって行う．ただし，ソフトウェア割込みハンドラでパケットの処理を行った場合には優先度逆転問題が発生することが知られている [2]．そこで LW-LBE では，ソフトウェア割込みハンドラを優先度ごとに作成し，ソフトウェア割込みハンドラとタスクをパケットの優先度順に実行するようにスケジューリングすることで優先度逆転を抑制する．LW-LBE を Android 端末上に実装して評価した結果，複数の通信が同時に行われたときのフォアグラウンド通信の遅延時間を抑制できるとともに，既存手法よりも少ない CPU サイクル数で実行できることが確認できた．

本稿の構成は以下のとおりである．2 章では，携帯電話センシングでは低優先度通信が必要なことと，既存の低優先度通信について述べる．3 章では，携帯電話上で低優先度通信を実現するために，CPU の負荷を抑制しつつパケットの優先度逆転を軽減する LW-LBE の設計を示す．4 章では，Android 端末上での LW-LBE の実装方法について述べる．5 章では，LW-LBE を実装した端末を使用して行った実験評価の結果を示す．6 章では，LW-LBE を適用するうえでの条件について議論する．7 章では，まとめを述べる．

2. 低優先度通信

携帯電話において，GPS やマイク，カメラ，加速度センサ，地磁気センサ，角速度センサなど多彩なセンサの搭載が進められていることを背景に，携帯電話を利用して実空間情報を収集する携帯電話センシングの研究が行われている [1]．個々のユーザがデータを収集してセンサネットワークを構成することから，Participatory Sensing [3] や People-Centric Sensing [4] と呼ばれている．すでに，環境モニタリング [5], [6], [7], [8], [9] やヘルスケア [10], [11]，ソーシャルネットワークでの利用 [12], [13], [14] などの研究が進められている．

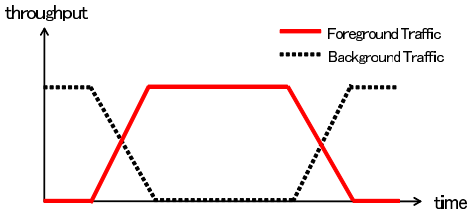

図 1 低優先度通信のスループット特性

Fig. 1 Throughput characteristics of lower-than-best-effort traffic.

携帯電話センシングでは，センサデータがバックグラウンドで送信されることが想定される．そのため，バックグラウンド通信によって既存のフォアグラウンドのアプリケーションの通信に遅延が生じることによる操作性の低下が問題になる．

フォアグラウンドのアプリケーションの通信遅延を抑えるためには，フォアグラウンドのアプリケーションのタスクとパケットを高優先度に設定する方法が考えられる．既存のアプリケーションをすべて書き換えるのは現実的ではないため，既存のアプリケーションを変更せずに外部からタスクとパケットを高優先度に設定する必要がある．しかしながら，個々のアプリケーションではすでに優先度が設定されていることもあるため，外部から既存のタスクの優先度を変更することはアプリケーションの挙動に影響を与える可能性がある．また，外部からパケットを高優先度に設定するためには，すべてのアプリケーションが通信に使用している IP アドレスとポート番号を抽出して高優先度に設定するという負担が生じる．

既存のフォアグラウンドアプリケーションに変更を加えずに携帯電話センシングアプリケーションを導入するためには，バックグラウンドの通信のみをフォアグラウンドの通信よりも低い優先度で行う低優先度通信 [15] が求められる．低優先度通信の動作例を**図 1** に示す．低優先度通信では他の通信が実行されている間はバックグラウンド通信を抑制し，実行されていない間は通常の通信を行うことによって，フォアグラウンド通信の遅延を抑制しつつ空いた帯域を有効活用する．

低優先度通信を実現する方法として，IP 層でのアプローチと TCP 層でのアプローチの 2 つが考えられる [16]．IP 層で低優先度通信を実現する方法としてプライオリティキューイング（Priority Queueing）[17] が存在する．プライオリティキューイングでは，インターネット上の各ルータにおいてキューを優先度別に分け，優先度の高いキューのパケットから順に処理することでトラヒックの優先制御

を実現する．たとえばDiffServでは，IPパケットのTOSフィールドに優先度クラスを設定することでルータ上でのパケットの優先制御を実現している．しかしながら，現在のインターネット上のルータでは必ずしもすべてが優先制御に対応しているわけではないため，低優先度通信にプライオリティキューイングを利用することはできない．

TCP層で低優先度通信を実現する方法としてはTCP Nice [18] やTCP-LP [19]，LEDBAT [20] などが存在する．LEDBATは実際にP2PツールであるBitTorrentで採用されている [15], [20]．TCP層を利用して，バックグラウンドのトラヒックをEnd-Endで制御することで，ルータに変更を加えることなく低優先度通信を実現することができる．たとえば，TCP Niceでは，データフレームとデータフレームに対するACKが返ってくるまでのラウンドトリップタイムを計測し，ラウンドトリップタイムが閾値を超えた場合にはウィンドウサイズを半減させることでバックグラウンドトラヒックのスループットを抑制する．しかしながら，TCP層のアプローチでは，フォアグラウンド通信の有無をラウンドトリップタイムによって判断するため，トラヒックの制御が開始されるまでの間にフィードバック遅延が生じるという問題がある．

3. Lightweight Lower-than-Best-Effort

2章で述べたように，低優先度通信をIP層で実現する場合にはルータの変更が必要になり，TCP層で実現する場合にはフィードバック遅延が生じる．ここでIP層でのプライオリティキューイングについて，携帯電話センシングを対象とした場合を考える．プライオリティキューイングが有効なのは，経路内の各ノードにおいて優先度の異なる多数のパケットがキューの中に滞留している場合である．携帯電話からの上りのトラヒックが多い携帯電話センシングを考えた場合，携帯電話と無線基地局をつなぐリンクが狭いため [21]，優先度の異なるパケットは携帯電話内のキューに滞留する．このような携帯電話センシングの特徴を考慮すると，携帯電話内でのみプライオリティキューイングを行うことで，ある程度は低優先度通信を実現することができると考えられる．

また，近年の携帯電話では，LinuxやSymbian，Androidなどの組込OSとしては高機能なOSを搭載している [22], [23]．すなわち，IP層でのトラヒックの優先制御を携帯電話上に実装する場合には，オペレーティングシステム上のソフトウェアとして実現される．オペレーティングシステムでは，タスクや割込みハンドラを組み合わせることで多様な処理を実現しているため，タスクや割込みハンドラの特性を考慮して低優先度通信機構を設計する必要がある．

以上の議論に鑑み，携帯電話においてIP層での低優先度通信を低オーバヘッドに実現する低優先度スケジューリング機構「LW-LBE（Lightweight Lower-than-Best-Effort）」の設計を行った．LW-LBEでは，ソフトウェア割込みハンドラでプライオリティキューのパケットの処理を行うと同時に，実行中のタスクに応じてソフトウェア割込みハンドラをスケジューリングすることで携帯電話上での低優先度通信を低オーバヘッドで実現する．

3.1 携帯電話上で優先制御を行う場合の問題点

携帯電話において優先制御を行う場合，ハードウェア送信キューにおけるパケットの優先度逆転問題と，プライオリティキューにおけるタスクの実行順序逆転問題が発生する．

ハードウェア送信キューにおけるパケットの優先度逆転問題は，ハードウェア送信キューに異なる優先度のパケットが保持されている場合でも，先に挿入された低優先度のパケットが先に送出されてしまうという問題である．ハードウェア送信キューのサイズを小さくすることでパケットの優先度逆転を抑えることができるが，送信キューが空になった場合に発生する送信キュー空き通知が増加するという問題が新たに発生する．

プライオリティキューにおけるタスクの実行順序逆転問題は，割込みハンドラとタスクの間で発生する問題である．携帯電話でも採用されているオペレーティングシステムでは，実行の制御単位であるタスクと割込みハンドラが独立したスケジューリング機構を持ち [24]，不定期に発生する外部イベントに反応して動作するために割込みハンドラがタスクよりも優先して実行される [25]．そのため，プライオリティキューのパケットをソフトウェア割込みハンドラで処理した場合には，高優先度パケットを処理するタスクと低優先度パケットを処理するソフトウェア割込みハンドラの間で実行順序の逆転が発生する．

図 2 に理想的な実行順序の例を示す．図2では，低優先度のアプリケーションがパケットを送出した直後に高優先度のアプリケーションがパケットを送出し，高優先度のパケットが先に処理されている．しかしながら，通常のオペレーティングシステムでは，パケット処理をタスクより実行優先度の高いソフトウェア割込みハンドラで行っているために実行順序の逆転が発生する．**図 3** に実行順序の逆転が発生した例を示す．図3では，高優先度のアプリケーションがパケットの送出を実行する前に無線通信モジュールからハードウェア送信キューが空いたという通知がハー

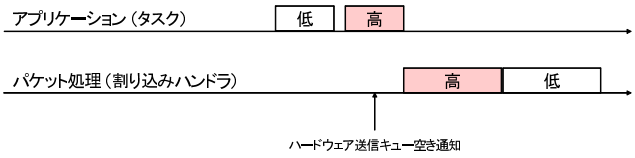

図 2 理想的な実行順序
Fig. 2 Ideal execution sequence.

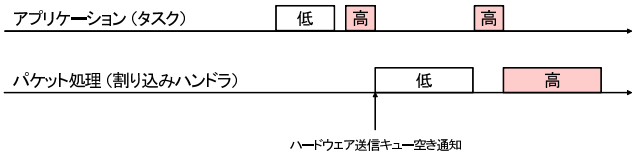

図 3 実行順序の逆転
Fig. 3 Actual execution sequence.

ドウェア割込みとして発生する．するとハードウェア割込みハンドラが，すでにプライオリティキューに挿入されている低優先度パケットを処理するためのソフトウェア割込みハンドラを起動し，高優先度のアプリケーションは中断される．これにより，低優先度のパケットが高優先度のパケットよりも先に処理されてしまう．

プライオリティキューにおけるタスクの実行順序逆転問題を解決する手法として，Tokuda らの研究 [2] や Lim らの研究 [26] が存在する．これらの研究では，プライオリティキューに格納されたパケットを，ソフトウェア割込みハンドラではなく，優先度ごとに作成したタスクで処理することで実行順序逆転を抑制している．しかしながら，一般的に，タスクでの処理はソフトウェア割込みハンドラでの処理よりも切替えのオーバヘッドが大きい [27]．たとえば，処理を切り替える際にオーバヘッドの大きいスケジューリングやコンテキストスイッチ [27], [28], [29]，イベント情報を通知するための IPC（Inter-Process Communication）[30], [31] がパケットの処理を行う場合には頻発する．そのため，先に述べたハードウェア送信キューを小さくした場合に発生する送信キュー空き通知の増加が計算資源を多く消費してしまう．

3.2 LW-LBE の設計

3.1 節で述べたパケットの優先度逆転問題とタスクの実行順序逆転問題を同時に解決する方法として，LW-LBE では，ソフトウェア割込みハンドラをスケジューリングするというアプローチをとる．設計に際し，フォアグラウンドのアプリケーションの通信時間が，バックグラウンドのアプリケーションがセンサデータをアップロードしていない状況と同じ通信時間を実現することを目指す．LW-LBE の全体像を**図 4** に示す．LW-LBE は低優先度通信タスク，優先度情報管理機構，割込み制御機構の 3 つから構成される．

まず，低優先度通信を行うタスク（低優先度通信タスク）は自身のタスクの実行優先度を低優先度に設定すると同時に，低優先度通信を行う宛先 IP アドレス，宛先ポート番号，送信元ポート番号を優先度情報管理機構に対して登録する．たとえば 4 章で述べる Linux 上での実装では，タスクの実行優先度をタスク内で `setpriority` 関数を用いて 139 に設定したものを低優先度通信タスクとする．送信元ポート番号は低優先度通信タスク内で `connect` 関数を呼んだ後に `getsockname` 関数を利用するなどして取得する．

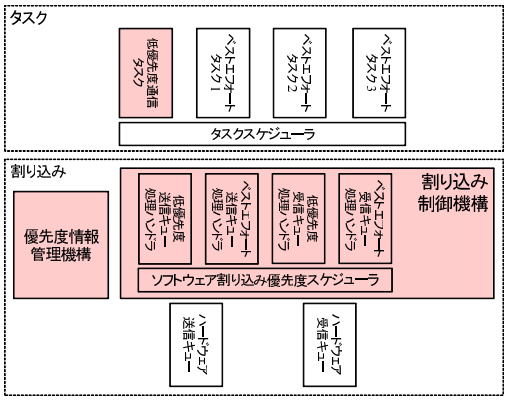

図 4 LW-LBE の全体像
Fig. 4 LW-LBE overview.

優先度情報管理機構に登録された IP アドレスとポート番号は，パケットの割り振りに利用する．タスクからパケットが送信された場合，優先度情報管理機構に登録されている IP アドレスとポート番号が含まれるパケットは低優先度送信キューに，登録されていないパケットはベストエフォート送信キューに割り振られる．パケットの受信時にも同様に，優先度情報管理機構に登録されている IP アドレスとポート番号のパケットは低優先度受信キューに，登録されていないパケットはベストエフォート受信キューに割り振られる．

割込み制御機構では，ハードウェア送信キューを 1 パケットの最大長まで最小化すると同時に，ソフトウェア割込みハンドラをスケジューリングすることで 3.1 節で述べた携帯電話上で優先制御を行う場合の優先度逆転問題と実行順序逆転問題を抑制する．

割込み制御機構は，ソフトウェア割込みスケジューラとプライオリティキューのパケットの処理を行う低優先度送信キュー処理ハンドラ，ベストエフォート送信キュー処理ハンドラ，低優先度受信キュー処理ハンドラ，ベストエフォート受信キュー処理ハンドラの 4 つのソフトウェア割込みハンドラから構成される．各ハンドラはそれぞれ優先度に応じたパケットを保持するキューを具備している．

ソフトウェア割込み優先度スケジューラは，3.1 節で述べた実行順序逆転問題を抑制するための機構である．**図 5** に，ソフトウェア割込み優先度スケジューラの処理手順を示す．ソフトウェア割込み優先度スケジューラは，ハードウェア割込みハンドラにおいてハードウェア割込みの処理が完了したタイミングと，タスクスケジューラにおいてタスクスイッチが発生したタイミングの 2 つのタイミングで呼び出される．

まず，ハードウェア割込みハンドラはハードウェア割込みが発生すると，ハードウェア割込みの解除やプライオリティキューへの受信パケットの取り込みを行った後に，ソフトウェア割込み優先度スケジューラを呼び出す．ソフトウェア割込み優先度スケジューラは，ベストエフォート

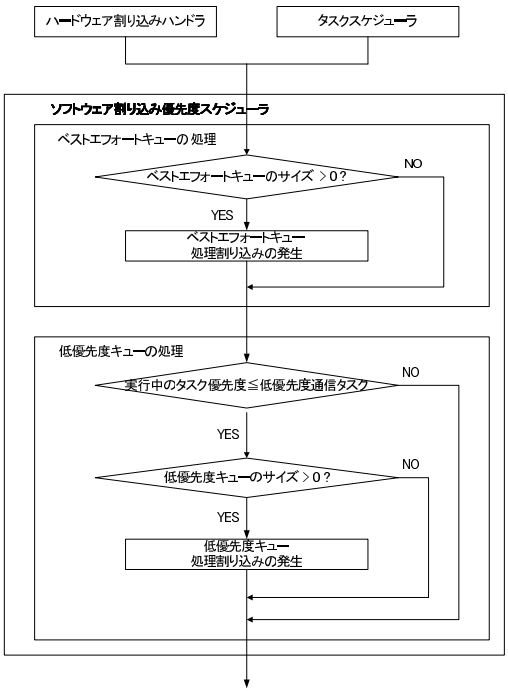

図 5 ソフトウェア割込み優先度スケジューラの処理手順
Fig. 5 Flow chart of software-interrupt scheduler.

キューにパケットが存在する場合には，ベストエフォートキュー処理ソフトウェア割込みを発生させる．実行中のタスクが低優先度通信タスクで，低優先度キューにパケットが存在する場合には，低優先度キュー処理ソフトウェア割込みを発生させる．たとえば 4 章で述べる Linux 上での実装では，カーネル内の `current` 変数を利用して実行中のタスクの優先度が 139 であった場合には低優先度通信タスクが実行されていると判定する．ベストエフォートキュー処理ソフトウェア割込みと低優先度キュー処理ソフトウェア割込みの発生を受けて，カーネルがすべてのソフトウェア割込みハンドラを実行する．

実行中のタスクが低優先度通信タスクでなかった場合には，ベストエフォートキュー処理ソフトウェア割込みのみを発生させる．低優先度キューにパケットが存在する場合であっても，低優先度キュー処理ソフトウェア割込みは発生しないため，カーネルは低優先度キュー処理ハンドラ以外のソフトウェア割込みハンドラを実行する．

ハードウェア割込みが発生したタイミングで起動が抑制されたソフトウェア割込みハンドラは，タスクスイッチが発生したタイミングで実行される．タスクスケジューラにおいてタスクスイッチが発生すると，実行中のタスクが切り替わるため，未実行の処理ハンドラを実行するためにソフトウェア割込み優先度スケジューラを呼び出す．

3.3 通信を行わないフォアグラウンドアプリケーションがあった場合の動作

LW-LBE では，フォアグラウンドがオフラインゲームなどの通信を行わないアプリケーションであっても，フォアグラウンドタスクの優先度が高ければ，バックグラウンド通信のためのソフトウェア割込みハンドラの起動を抑制する．すなわち，通信資源が枯渇していない場合でもバックグラウンド通信のための処理を遅滞させる．

CPU 使用率が常時 100%でない場合には，CPU に空きが出たタイミングでソフトウェア割込みハンドラを起動するため，バックグラウンドの通信は著しく非効率な通信とはならない．CPU 使用率が常時 100%である場合には，バックグラウンド通信の処理が長時間実行されなくなるため，バックグラウンドの通信は非効率な通信となる．しかしながら，バックグラウンド通信のための処理を遅滞させなければ，優先度の高いフォアグラウンドタスクの処理が中断されてユーザの操作性が低下するため，LW-LBE による優先度制御は必要であると考えている．

4. 実装

オペレーティングシステムとして Linux カーネルを使用する Android 端末上で LW-LBE の実装を行った．Android はバージョン 2.0，Linux カーネルはバージョン 2.6.29，Android 端末として GDD フォンを用いた．なお，本稿では Linux 上での実装を行ったが，Symbian，iOS，Windows Phone などの携帯電話向けのオペレーティングシステム [22] にも実装可能であると考えている．

4.1 ハードウェア送信キューの最小化

ハードウェア送信キューの最小化はネットワークドライバに手を加えることで実現した．ネットワークドライバにおいて，バッファ内のデータサイズが閾値 th を超えると Linux カーネルで提供されている `netif_stop_queue()` を呼び出してハードウェアバッファへの書き込みを禁止し，th 以下になると `netif_wake_queue()` を呼び出してハードウェアバッファへの書き込みを許可するようにした．th の値は，パケットサイズの上限である MTU の値 1,500 byte に合わせて 1,500 byte とした．

4.2 プライオリティキュー処理ハンドラ

各プライオリティキュー処理ハンドラはそれぞれパケットを保持するプライオリティキューを備えている．送信用のプライオリティキューは，Linux カーネルの機能を利用して実現した．コンフィギュレーションにより “QoS and/or fair queueing” と “Multi Band Priority Queueing (PRIQ)” を有効化し，`tc qdisc` コマンドを実行して低優先度送信キューとベストエフォート送信キューを作成した．受信用のプライオリティキューは，受信パケットのリストを表す `input_pkt_queue` を優先度ごとに設けることによって実現した．

プライオリティキューに格納されたパケットは，優先度

順に処理が実行されるように，優先度ごとに分割したソフトウェア割込みハンドラを使って処理される．Linux カーネルにおける送信用ソフトウェア割込みハンドラである net_tx_action() と受信用ソフトウェア割込みハンドラである net_rx _action() をプライオリティキューごとに割り当てた．

4.3 ソフトウェア割込み優先度スケジューラ

ソフトウェア割込み優先度スケジューラはソフトウェア割込みハンドラの起動を行う raise_softirq_irqoff()，タスクのスケジューリングを行う schedule() の 2 つの関数を変更することで実現した．raise_softirq_irqoff() では，プライオリティキューにパケットが存在した場合に，実行中のタスクの優先度に応じてどのプライオリティキュー処理ハンドラを実行するかのスケジューリングを行う．raise_softirq_irqoff() の中で，実行中のタスク優先度 current->static_prio と低優先度タスクの優先度を意味する 139 とを比較し，current->static_prio の方が高ければ，ベストエフォート送信キュー処理ハンドラとベストエフォート受信キューハンドラのみを起動する．また，schedule() からも raise_softirq_irqoff() を呼び出すことで，タスクスケジューラからもソフトウェア割込みハンドラがスケジューリングされるようにした．

4.4 ネットワーク以外の割込みハンドラに対する影響

LW-LBE では，ソフトウェア割込みハンドラに手を加えるため，他の割込みハンドラに与える影響を考慮する必要がある．具体的には，ハードウェア割込みハンドラ，ネットワークのソフトウェア割込みハンドラよりも優先度の高いソフトウェア割込みハンドラ，ネットワークのソフトウェア割込みハンドラよりも優先度の低いソフトウェア割込みハンドラの 3 つについて考慮する必要がある．

ハードウェア割込みハンドラは，ソフトウェア割込みハンドラよりも優先して実行されるため，LW-LBE の制御による影響は受けないと考えられる．タイマなどネットワークのソフトウェア割込みハンドラよりも優先度の高いソフトウェア割込みハンドラも，ハードウェア割込みハンドラと同じく，LW-LBE の制御による影響は受けないと考えられる．タスクレットなどネットワークのソフトウェア割込みハンドラよりも優先度の低いソフトウェア割込みハンドラは，低優先度のネットワークソフトウェア割込みハンドラが起動されなくなることで，応答時間が早くなる可能性がある．いずれの場合も LW-LBE を適用することによる処理遅延が発生する可能性は低いと考えられる．GDD フォンで確認したところ．タッチやキー操作イベントを処理する際にソフトウェア割込みハンドラは使用されておらず，LW-LBE が動作している環境下でもタッチやキー操作時に瞬間的に停止するなどの異常は確認されなかった．

5. 評価

4 章に示した Android 端末上に実装した LW-LBE を用いて，フォアグラウンド通信の遅延時間，トラヒックの優先制御を行っている間の CPU 負荷について評価を行った．

5.1 評価環境

開発用 Android 端末である GDD フォンを使用して評価を行った．Android のバージョンは 2.0，Linux カーネルは 2.6.29 である．CPU はコアとして ARM11 を具備した Qualcomm MSM7200A 528 MHz，メモリは ROM が 512 MB，RAM が 192 MB である．低優先度通信の性能を相対的に評価するために，次の 4 手法を用いた．

(1) バックグラウンドトラヒックなし（Without Background Traffic）
この手法はバックグラウンドの通信を行わずに，フォアグラウンドのトラヒックの計測を行ったものである．他手法の性能を測るためのベースラインとなる．バックグラウンド通信がないため，最も性能が良いことが予想される．ハードウェア送信キューのサイズは，GDD フォンにおけるバッファサイズの最大値が 8 KB であったことから 8 KB とした．

(2) 低優先度通信なし（Without Lower-than-Best-Effort）
この手法はバックグラウンドでトラヒックを発生させた中で，低優先度通信を行わずにフォアグラウンドのトラヒックの計測を行ったものである．通常の携帯電話に何も手を加えていない状態を想定している．4 手法の中で遅延が最も大きいと予想される．ハードウェア送信キューのサイズは，GDD フォンにおけるバッファサイズの最大値である 8 KB とした．

(3) タスクによる低優先度通信（Task-based Lower-than-Best-Effort）
この手法はバックグラウンドでトラヒックを発生させた中で，タスクを用いて実装した低優先度通信でフォアグラウンドのトラヒックの計測を行ったものである．Tokuda らの研究 [2] における優先制御手法を想定して実装した．ハードウェア送信キューのサイズは，パケットの優先度逆転を抑制するためにパケットサイズの上限値である 1,500 byte とした．

(4) Lightweight Lower-than-Best-Effort（LW-LBE）
この手法は 3 章に示した提案手法である．バックグラウンドでトラヒックを発生させた中で，LW-LBE を用いて実装した低優先度通信でフォアグラウンドのトラヒックの計測を行ったものである．ハードウェア送信キューのサイズは，パケットの優先度逆転を抑制するためにパケットサイズの上限値である 1,500 byte とした．

表 1 割込みハンドラとタスクのオーバヘッドの比較
Table 1 Comparison of overhead between interrupt handler and task.

タスクと割込みハンドラの切替え	タスクとタスクの切替え
2,177 Cycle	5,484 Cycle

5.2 割込みハンドラとタスクのオーバヘッドの比較

割込みハンドラを用いた場合に，タスクに比べてオーバヘッドが削減できているかどうかを確認するために，割込みハンドラとタスクの CPU の実行サイクル数の比較を行った．具体的には，タスクとソフトウェア割込みハンドラの切替え，タスクとタスクの切替えをそれぞれ 3,000 回ずつ行い，1 回の切替えに要した実行サイクル数を算出した．実行サイクル数は ARM コアが具備する Performance Monitor [32] の機能を用いて取得した．

表 1 に 1 回あたりの実行サイクル数を示す．タスクと割込みハンドラを切り替えた場合，1 回あたりに平均 2,177 サイクルを要した．一方でタスクとタスクを切り替えた場合には，1 回あたり平均 5,484 サイクルを要した．すなわち，割込みハンドラの方がタスクよりも約 60%少ない実行サイクル数で起動ができることが確認された．

5.3 通信時間と実行サイクル数の評価

低優先度通信が低オーバヘッドで実現できているかどうかを確認するために，上りのバックグラウンドトラヒックが発生しているときに，転送するデータサイズを 1 KB から 1,000 KB まで変化させて，ダウンロード時間，アップロード時間，実行サイクル数の計測を行った．バックグラウンドではセンシングデータをアップロードしていることを模している．フォアグラウンドでは，ダウンロード時間の評価はウェブの閲覧をしている場合を，アップロード時間の評価はウェブへの投稿やメールの送信をしている場合を想定している．通信プロトコルは HTTP を使用し，ダウンロードには GET，アップロードには POST の 1 回のリクエストを実行することによって，すべてのデータを転送する．ダウンロード・アップロード時間と実行サイクル数は，それぞれ，フォアグラウンドのアプリケーションがダウンロード・アップロードを開始してからダウンロード・アップロードを終了するまでの時間と実行サイクル数である．通信環境としては W-CDMA のネットワークを用いた．バックグラウンドトラヒックとしては，携帯電話センシングで帯域を最大限に利用していることを想定して，バックグラウンドのアプリケーションが疑似データを低優先度で連続送信する．また，フォアグラウンドとバックグラウンドの通信はともに国内の 2 つの異なる地点のサーバを用いた．携帯電話ネットワークでは，セル内に存在するユーザの数や無線通信環境の変化によって，通信時間が変動する．本稿では，ネットワーク環境による影響を排除するために，測定結果として 100 回の平均値を使用した．

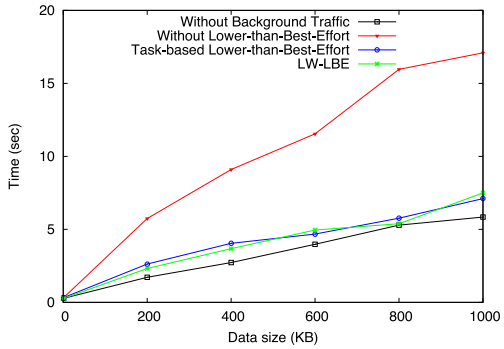

図 6 フォアグラウンドがダウンロード，バックグラウンドがアップロードの場合のフォアグラウンドのダウンロード時間
Fig. 6 Download time while a foreground application is downloading data and a background application is uploading data.

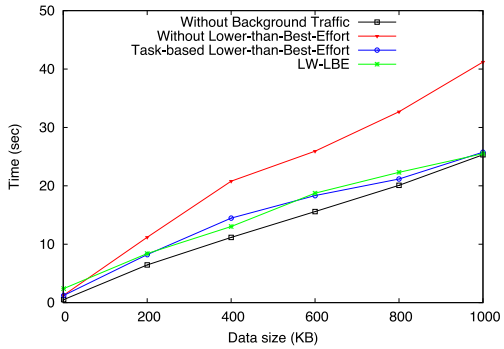

図 7 フォアグラウンドがアップロード，バックグラウンドがアップロードの場合のフォアグラウンドのアップロード時間
Fig. 7 Upload time while a foreground application is uploading data and a background application is uploading data.

任意のサイズのデータに対するダウンロード時間を**図 6** に，アップロード時間を**図 7** に示す．図の横軸はフォアグラウンドでダウンロード・アップロードしたデータのサイズ，縦軸はフォアグラウンドのダウンロード・アップロード時間である．図 6 と図 7 より次の 3 つのことが分かる．

(1) LW-LBE は低優先度通信なし（Without Lower-than-Best-Effort）よりもダウンロード・アップロード時間が短い．これは LW-LBE において，フォアグラウンドのダウンロード・アップロードパケットの方がバックグラウンドのアップロードパケットよりも優先的に制御されているからだと考えられる．

(2) LW-LBE とタスクによる低優先度通信（Task-based Lower-than-Best-Effort）とがほぼ同等のダウンロード・アップロード時間を実現している．たとえばデータサイズが 400 KB のときのダウンロード時間は，LW-LBE が約 3.68 秒，タスクによる低優先度通信が約 4.04 秒である．

(3) LW-LBE がバックグラウンドトラヒックなし（Without Background Traffic）よりもダウンロード・アッ

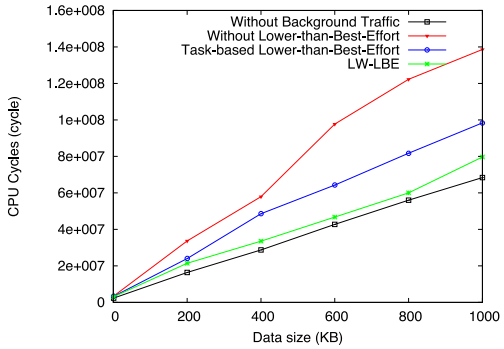

図 8 フォアグラウンドがダウンロード，バックグラウンドがアップロードの場合の実行サイクル数

Fig. 8 Execution cycles while a foreground application is downloading data and a background application is uploading data.

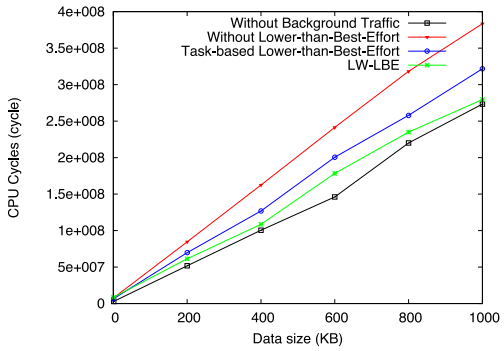

図 9 フォアグラウンドがアップロード，バックグラウンドがアップロードの場合の実行サイクル数

Fig. 9 Execution cycles while a foreground application is uploading data and a background application is uploading data.

プロード時間が長い．これは，バックグラウンドトラヒックなしではフォアグラウンドトラヒックのハードウェア割込みのみが発生するのに対し，LW-LBE では，フォアグウランドトラヒックとバックグラウンドトラヒックの両方のハードウェア割込みが発生するからだと考えられる．

任意のサイズのデータに対する実行サイクル数を**図 8** と**図 9** に示す．図の横軸はフォアグラウンドでダウンロード・アップロードしたデータのサイズ，縦軸はフォアグラウンドでダウンロード・アップロードを開始してから終了するまでの実行サイクル数である．図 8 と図 9 より次の 3 つのことが分かる．

(1) LW-LBE は，タスクによる低優先度通信（Task-based Lower-than-Best-Effort）よりも実行サイクル数が小さい．これは，LW-LBE ではタスクよりもオーバヘッドの小さいソフトウェア割込みハンドラを用いて低優先度通信を実現しているからだと考えられる．

(2) LW-LBE の実行サイクル数が，バックグラウンドトラヒックなし（Without Background Traffic）の実行サイクル数よりも多い．これは，前述のダウンロード・アップロード時間の計測と同様に，バックグラウンドトラヒックなしではフォアグラウンドトラヒックのハードウェア割込みのみが発生するのに対し，LW-LBE の計測では，バックグラウンドトラヒックとフォアグラウンドトラヒックの両方のハードウェア割込みが発生するからだと考えられる．

(3) 低優先度通信を用いない場合（Without Lower-than-Best-Effort）には，実行サイクル数が最も多い．これは，低優先度通信を用いていないことにより，フォアグラウンド通信でのダウンロード・アップロード時間が長くなり，結果としてバックグラウンド通信にも多くの計算資源を消費してしまうからだと考えられる．

以上より，携帯電話センシングを対象としたトラヒックでは，LW-LBE は低オーバヘッドで低優先度通信を実現していることが分かった．実行サイクル数が削減されることにより，通信に関わる処理が少なくなるため，アプリケーションの応答性を高めることができる．また，CPU が高優先度のタスクによって高負荷の状態では，実行サイクル数の削減は通信処理の効率化につながるため，少ない CPU 空き時間に低優先度通信の処理を完了でき，結果として消費電力の大きい無線通信の時間を減らすことができると考えている．さらに，CPU が低負荷の状態でも，わずかであるものの実行サイクル数を削減した分だけ CPU のスリープ時間が増加するため，消費電力を下げられると考えている．これらの効果は，通信を行えば行うほど顕著になってゆくと予想される．

本稿では，携帯電話センシングのアプリケーションがバックグラウンドで通信を行ったときに，フォアグラウンドのアプリケーションが単独で通信を行った場合と同程度まで通信遅延が抑制できることを目標とした．評価の結果，既存手法と同程度までフォアグラウンド通信の遅延を抑制することはできたものの，フォアグラウンドアプリケーションが単独で通信を行った場合と同じ通信時間を実現することはできなかった．この要因として，フォアグラウンドアプリケーションの処理中に，バックグラウンドアプリケーションのためのハードウェア割込みを抑制することができず，フォアグラウンドの処理が中断されてしまったことが考えられる．

ハードウェア割込みによる影響を抑制する方法として，アプリケーションプロセッサをマルチコア化する方法が考えられる．マルチコア化することで，バックグラウンドアプリケーションのためのハードウェア割込みが別のプロセッサで処理され，フォアグラウンドアプリケーションの遅延をさらに抑制することができると予想される．

このような観点から，LW-LBE のマルチコア環境での動作について考える．マルチコア環境下で同時実行可能な処理数はコア数までである．実行待ちの処理数がコア数以

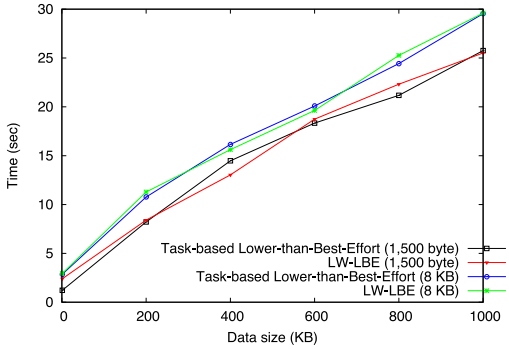

図 10 ハードウェア送信キューサイズを変化させたときのアップロード時間

Fig. 10 Upload time with different hardware queue sizes.

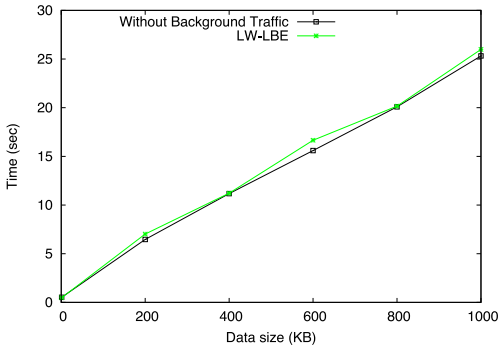

図 11 ハードウェア送信キュー最小化したときの通信オーバヘッドの評価（通信時間）

Fig. 11 Overhead of upload time in minimal hardware queue size.

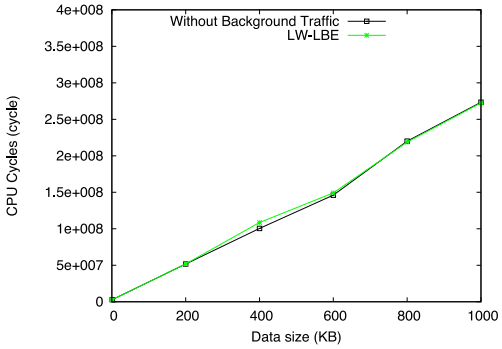

図 12 ハードウェア送信キュー最小化したときの通信オーバヘッドの評価（実行サイクル数）

Fig. 12 Overhead of execution cycles in minimal hardware queue size.

下の場合には，バックグラウンドアプリケーションの処理とフォアグラウンドアプリケーションの処理が異なるコアで同時に実行される．しかしながら，キューに入っているパケットを優先度に応じてハードウェアに書き込む処理は，ハードウェアキューに空きができたタイミングでソフトウェア割込みとして 1 つのコア内で実行されるため，LW-LBE による低優先度通信は実現される．

実行待ちの処理数がコア数より大きい場合には，LW-LBE はマルチコア環境でも現在の仕組みのままでタスクと割込み間の実行順序を制御する．すなわち，バックグラウンドアプリケーションの処理は抑制されるため，実行待ちの処理数がコア数よりも大きい場合でも低優先度通信が実現される．

5.4 ハードウェア送信キューサイズに関する評価

ハードウェア送信キューのサイズが大きくなると，高優先度パケット送信時に低優先度パケットがキューイングされている確率が増大するために，高優先度の通信に遅延が生じる．**図 10** に，タスクによる低優先度通信（Task-based Lower-than-Best-Effort）と Lightweight Lower-than-Best-Effort（LW-LBE）において，ハードウェア送信キューのサイズを 1,500 byte と 8 KB に設定したときのアップロード時間を示す．バックグラウンドでアプリケーションが疑似データを連続送信している状況で，フォアグラウンドでアプリケーションが任意サイズのデータをアップロードしている．図の横軸はフォアグラウンドでアップロードしたデータのサイズ，縦軸はフォアグラウンドのアップロード時間である．図 10 より，タスクによる低優先度通信と LW-LBE ともに，ハードウェア送信キューのサイズを 8 KB とした場合ではハードウェア送信キューのサイズが 1,500 byte の場合よりもフォアグラウンドの通信遅延が大きいことが分かる．ハードウェア送信キューのサイズを縮小することによって，フォアグラウンド通信の遅延を抑制することができていると考えられる．

ハードウェア送信キューのサイズを最小化することで，複数のアプリケーションが同時に通信を行う場合にはフォアグラウンド通信の遅延を抑制できるが，同時に通信を行わない場合には通信オーバヘッドの上昇を招くことが想定される．そこで，ハードウェア送信キューサイズを最小化したときに生じる通信オーバヘッドについて評価を行った．**図 11** に，バックグラウンドでのアップロードがない状態で，フォアグラウンドでアップロードを行ったときのアップロード時間を示す．図の横軸はフォアグラウンドでアップロードしたデータのサイズ，縦軸はフォアグラウンドのアップロード時間である．また，**図 12** に，バックグラウンドでのセンサデータのアップロードがない状態で，フォアグラウンドでアップロードを行ったときの実行サイクル数を示す．図の横軸はフォアグラウンドでアップロードしたデータのサイズ，縦軸はフォアグラウンドでアップロードを開始してから終了するまでの実行サイクル数である．図 11，図 12 ともに，バックグラウンドトラヒックなし（Without Background Traffic）と Lightweight Lower-than-Best-Effort（LW-LBE）で比較を行い，ハードウェア送信キューのサイズはバックグラウンドトラヒックなしが 8 KB，LW-LBE が 1,500 byte である．図 11 と図 12 より，バックグラウンドトラヒックなしと LW-LBE

で，通信時間および実行サイクル数に大きな違いが生じていないことが分かる．ハードウェア送信キューを最小化することでハードウェア送信キューへの書き込み頻度は増大するものの，ソフトウェア割込みハンドラの実行オーバヘッドが小さいために，通信時間，実行サイクル数ともに影響は限定的であると考えられる．

5.5 フィードバック遅延に関する評価

TCP 層による低優先度通信機構を利用した場合に生じるフィードバック遅延が，フォアグラウンド通信の遅延に及ぼす影響について，TCP Nice [18] を例に考察を行う．

TCP Nice では，RTT が閾値を超えると輻輳ウィンドウサイズを半減する．ここでは，TCP Nice によって輻輳ウィンドウサイズが 1 になるまでの時間をフィードバック遅延と見なす．輻輳ウィンドウサイズが 1 になるまでに必要な制御回数を n，バックグラウンド通信の輻輳ウィンドウサイズを w とすると，w を n 回半減するため，

$$\frac{w}{2^n} \simeq 1 \tag{1}$$

が成り立つ．式 (1) より，

$$n \simeq \lceil \log_2 w \rceil \tag{2}$$

を導くことができる．ここで，$\lceil x \rceil$ は実数 x に対して x 以上の最小の整数を表す．RTT の値は，接続先サーバから ACK が返されたときに更新される．現在の TCP では遅延 ACK が用いられており，サーバにおいて送信データが発生するか，遅延確認応答タイマがタイムアウトしたときに ACK が返される [33]．携帯電話センシングのように連続してデータを送信する環境下では，サーバからデータが送信される頻度は少なく，遅延確認応答タイマがタイムアウトしたときに ACK が返される．遅延確認応答タイマのタイムアウト間隔を T_{delay} とすると，輻輳ウィンドウサイズが 1 になるまでのフィードバック遅延時間 $t_{feedback}$ は，遅延確認応答タイマがタイムアウトする時間 T_{delay} の n 回分となるため，

$$t_{feedback} = nT_{delay} \tag{3}$$

と表される．

W-CDMA 環境下で GDD フォンを使って 100 回計測を行った結果，連続してデータをアップロードしたときの輻輳ウィンドウサイズ w は 65 であった．遅延確認応答タイマ T_{delay} の値は 200 ms と設定されることが多いことから，式 (2)，(3) より $t_{feedback} \simeq 1{,}400$ msec と求められる．1,400 msec は人間にとって知覚可能な時間であり，現時点では TCP 層による低優先度通信を利用した場合のフィードバック遅延を無視できないと考えている．

6. LW-LBE の適用領域に関する議論

携帯電話上で発生するバックグラウンド通信については，携帯電話センシング以外にも，カレンダアプリケーションやメーラなど既存のアプリケーションによる通信が存在する．LW-LBE では，事前にバックグラウンドのアプリケーションが宛先 IP アドレスと宛先ポート番号，送信元ポート番号を登録しておく必要があるため，既存のバックグラウンド通信に対しては適用することができず，フォアグラウンドの通信に遅延が生じる．

しかしながら，近年リリースされた携帯電話ソフトウェア基盤の機能を利用することで，既存のバックグラウンド通信に対しても低優先度通信機構を適用可能である．スマートフォンでは，バックグラウンド通信による通信料金の増大やバッテリの消耗が問題となっており，アプリケーションがフォアグラウンドにあるかバックグラウンドにあるかを管理するソフトウェア基盤が現れている．たとえば，2011 年 11 月にリリースされた Android 4.0 では，Activity Manager と呼ばれるアプリケーション管理機構でアプリケーションがフォアグラウンドにあるかバックグラウンドにあるかを管理することによって，バックグラウンド通信を遮断する機能を提供している．アプリケーション管理機能を利用してアプリケーションがフォアグラウンドにあるかバックグラウンドにあるかを識別し，ソケットレベルで宛先 IP アドレスと宛先ポート番号，送信元ポート番号を取得すれば，事前の登録を必要とせずに低優先度通信機構を実現することができる．

ただし，LW-LBE では，タスクとパケットを優先度順に処理することで高優先度トラヒックの遅延を抑制するため，優先度に応じてタスクとパケットを分離できる必要がある．たとえば，1 つのタスクが同一のサーバに高優先度のパケットと低優先度のパケットの 2 種類を送信する場合には，優先度に応じて実行順序を制御することができない．したがって，LW-LBE を利用する場合には，タスクを優先度ごとに分割し，送信先あるいは送信元のポート番号を分けるなどする必要がある．

また，LW-LBE で用いているソフトウェア割込みハンドラを優先度ごとに分割し，ソフトウェア割込みハンドラとタスクの間の実行順序を制御して優先度逆転を抑制する機構は，ネットワーク以外のソフトウェア割込みハンドラに対しても適用できる．たとえば，タスクレットで処理される共有の資源に対して，高優先度タスクと低優先度タスクがアクセスする場合には，優先度ごとに分割したタスクレットとタスクの間の実行順序を制御することで処理遅延を削減することができる．

7. おわりに

本稿では，携帯電話センシングを対象とした携帯電話において，フォアグランドのアプリケーションに影響を与えないことを目的として，低オーバヘッドで低優先度通信を実現する LW-LBE（Lightweight Lower-than-Best-Effort）

について述べた．LW-LBE は IP 層での優先度通信をソフトウェア割込みハンドラを用いて実現する．実行中のタスクの優先度に応じてソフトウェア割込みハンドラをスケジューリングすることで低オーバヘッド性を実現している．Android 端末上に LW-LBE を含む 4 手法を実装し，通信時間と実行サイクル数に関する実験評価を行った．その結果，通信時間に関しては LW-LBE は既存の手法と同等の性能，実行サイクル数に関しては既存の手法よりも小さいとの結論が得られた．

参考文献

[1] Lane, N.D., Miluzzo, E., Hong, L., Peebles, D., Choudhury, T. and Campbell, A.T.: A survey of mobile phone sensing, *IEEE Communications Magazine*, Vol.48, No.9, pp.140–150 (2010).

[2] Tokuda, H., Mercer, C.W., Ishikawa, Y. and Marchok, T.E.: Priority inversions in real-time communication, *IEEE RTSS*, pp.348–359 (1989).

[3] Burke, J., Estrin, D., Hansen, M., Parker, A., Ramanathan, N., Reddy, S. and Srivastava, M.B.: Participatory sensing, *ACM SenSys World Sensor Web Workshop* (2006).

[4] Campbell, A.T., Eisenman, S.B., Lane, N.D., Miluzzo, E., Peterson, R.A., Hong, L., Xiao, Z., Musolesi, M., Fodor, K. and Gahng-Seop, A.: The Rise of People-Centric Sensing, *IEEE Internet Computing*, Vol.12, No.8, pp.12–21 (2008).

[5] Mun, M., Reddy, S., Shilton, K., Yau, N., Burke, J., Estrin, D., Hansen, M., Howard, E., West, R. and Boda, P.: PEIR, the personal environmental impact report, as a platform for participatory sensing systems research, *ACM MobiSys*, pp.55–68 (2009).

[6] Maisonneuve, N., Stevens, M., Niessen, M.E. and Steels, L.: Noisetube: Measuring and mapping noise pollution with mobile phones, *ITEE*, pp.215–228 (2009).

[7] Lu, H., Pan, W., Lane, N.D., Choudhury, T. and Campbell, A.T.: SoundSense: scalable sound sensing for people-centric applications on mobile phones, *ACM MobiSys*, pp.165–178 (2009).

[8] Mohan, P., Padmanabhan, V.N. and Ramjee, R.: Nericell: Rich monitoring of road and traffic conditions using mobile smartphones, *ACM SenSys*, pp.323–336 (2008).

[9] Thiagarajan, A., Ravindranath, L., LaCurts, K., Madden, S., Balakrishnan, H., Toledo, S. and Eriksson, J.: VTrack: Accurate, energy-aware road traffic delay estimation using mobile phoness, *ACM SenSys*, pp.85–98 (2009).

[10] Consolvo, S., McDonald, D.W., Toscos, T., Chen, M.Y., Froehlich, J., Harrison, B., Klasnja, P., LaMarca, A., LeGrand, L., Libby, R., Smith, I. and Landay, J.A.: Activity Sensing in the Wild: A Field Trial of UbiFit Garden, *ACM SIGCHI*, pp.1797–1806 (2008).

[11] Dahlman, E., Ekstrom, H., Furuskar, A., Jading, Y., Karlsson, J., Lundevall, M. and Parkvall, S.: HealthAware: Tackling obesity with health aware smart phone systems, *IEEE ROBIO*, pp.1549–1554 (2009).

[12] Miluzzo, E., Lane, N.D., Fodor, K., Peterson, R., Lu, H., Musolesi, M., Eisenman, S.B., Zheng, X. and Campbell, A.T.: Sensing meets mobile social networks: The design, implementation and evaluation of the CenceMe application, *ACM SenSys*, pp.337–350 (2008).

[13] Musolesi, M., Miluzzo, E., Lane, N.D., Eisenman, S.B., Choudhury, T. and Campbell, A.T.: The Second Life of a Sensor Integrating Real-World Experience in Virtual Worlds using Mobile Phones, *HotEmNets* (2008).

[14] Gaonkar, S., Li, J., Choudhury, R.R., Cox, L. and Schmidt, A.: Micro-Blog: Sharing and querying content through mobile phones and social participation, *ACM MobiSys* (2008).

[15] Carofiglio, G., Muscariello, L., Rossi, D. and Testa, C.: A hands-on assessment of transport protocols with lower than best effort priority, *IEEE LCN*, pp.8–15 (2010).

[16] Kokku, R., Bohra, A., Ganguly, S. and Venkataramani, A.: A Multipath Background Network Architecture, *IEEE INFOCOM*, pp.1352–1360 (2007).

[17] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z. and Weiss, W.: RFC2475: An Architecture for Differentiated Services (1998).

[18] Venkataramani, A., Kokku, R. and Dahlin, M.: TCP Nice: A mechanism for background transfers, *ACM SIGOPS*, Vol.36, pp.329–343 (2002).

[19] Kuzmanovic, A. and Knightly, E.W.: TCP-LP: A distributed algorithm for low priority data transfer, *IEEE INFOCOM*, pp.1691–1701 (2003).

[20] Rossi, D., Testa, C., Valenti, S. and Muscariello, L.: LEDBAT: The New BitTorrent Congestion Control Protocol, *IEEE ICCCN*, pp.1–6 (2010).

[21] Liedtke, J.: Pervasive data access in wireless and mobile computing environments, *Wireless Communications and Mobile Computing*, Vol.8, pp.25–44 (2008).

[22] Tarkoma, S. and Lagerspetz, E.: Arching over the Mobile Computing Chasm: Platforms and Runtimes, *IEEE Computer*, Vol.44, No.4, pp.22–28 (2011).

[23] Gartner: Forecast: Mobile Communications Devices by Open Operating System, 2007-2014, available from ⟨http://www.gartner.com/it/page.jsp?id=1434613⟩.

[24] Zhang, Y. and West, R.: Process-Aware Interrupt Scheduling and Accounting, *IEEE RTSS*, pp.191–201 (2006).

[25] Leyva-del-Foyo, L.E., Mejia-Alvarez, P. and de Niz, D.: Predictable interrupt management for real time kernels over conventional PC hardware, *IEEE RTAS*, pp.14–23 (2006).

[26] Lim, H., Park, D., Kang, S. and Oh, B.: Priority queue-based IEEE1394 device driver supporting real-time characteristics, *IEEE Trans. Consumer Electronics*, Vol.46, No.3, pp.825–833 (2000).

[27] David, F.M., Carlyle, J. and Campbell, R.H.: Context switch overheads for Linux on ARM platforms, *ACM ExpCS* (2007).

[28] Tan, T.K., Raghunathan, A. and Jha, N.K.: Energy macromodeling of embedded operating systems, *ACM TECS*, Vol.4, No.1 (2005).

[29] Mogul, J.C. and Borg, A.: The effect of context switches on cache performance, *ACM SIGPLAN*, Vol.26, No.4, pp.75–84 (1991).

[30] Liedtke, J.: On micro-kernel construction, *ACM SIGOPS*, Vol.29, No.5 (1995).

[31] Parmer, G. and West, R.: Predictable Interrupt Management and Scheduling in the Composite Component-Based System, *IEEE RTAS*, pp.232–243 (2008).

[32] ARM11 performance monitor unit (2008). Application Note 195.

[33] Stevens, W.: *TCP/IP Illustrated Volume 1 The Protocols*, Addison-Wesley (1999).

推薦文

本稿は，低優先度通信におけるパケットの優先度逆転の問題を電力消費を抑えた形で解決することで，ユーザが使用するアプリケーションによるフォアグラウンド通信の遅延を抑制する手法を提案している．関連研究を十分把握し，タスク切替えに基づく既存方式の問題点をとらえて，ソフトウェア割込みを優先キューに割り当てる新規の実用的な解決法を提案している．提案手法は Android 実機上の Linux カーネルに実装されている．提案手法の有効性は既存方式と比較して通信速度，消費電力削減という 2 つの評価軸を用いて実証されている．また論文全体としても設定課題の明確さを維持しつつ分かりやすく記述され，高く評価できる．

（モバイルコンピューティングとユビキタス通信研究会
主査　竹下　敦）

山本 享弘　（学生会員）

平成 10 年京都大学工学部電気工学第二学科卒業．平成 12 年京都大学大学院工学研究科電気工学専攻修士課程修了．同年（株）エヌ・ティ・ティ・ドコモ入社．平成 17 年（株）コア入社．組込みソフトウェアの研究開発に従事．現在，東京大学先端科学技術研究センター博士課程に在籍．電子情報通信学会会員．

猿渡 俊介　（正会員）

平成 19 年東京大学大学院博士課程修了．平成 15～16 年 IPA 未踏ソフトウェア創造事業，平成 18～20 年日本学術振興会学振特別研究員，平成 19～20 年イリノイ大学客員研究員，現在，東京大学先端科学技術研究センター助教．専門はワイヤレスネットワーク，センサネットワーク，システムソフトウェア等．平成 21 年電子情報通信学会論文賞．平成 22 年情報処理学会山下記念研究賞．電子情報通信学会，IEEE，ACM 各会員．

森川 博之　（正会員）

昭和 62 年東京大学工学部電子工学科卒業．平成 4 年東京大学大学院博士課程修了．現在，東京大学先端科学技術研究センター教授．工学博士．平成 9～10 年コロンビア大学客員研究員．平成 14～18 年情報通信研究機構モバイルネットワークグループリーダ兼務．ユビキタスネットワーク，無線ネットワーク，モバイルコンピューティング，フォトニックインターネット等の研究に従事．本会論文賞，電子情報通信学会論文賞（3 回），ドコモモバイルサイエンス賞，志田林三郎賞，情報通信功績賞等受賞．電子情報通信学会フェロー，IEEE，ACM，ISOC，映像情報メディア学会各会員．