

モバイル端末における低優先度通信のための 低負荷パケットスケジューリング方式

山本 享弘^{†1,†2} 猿渡 俊介^{†1} 森川 博之^{†1}

近年の携帯電話の高機能化に伴い、複数のアプリケーションが同時に通信を行う環境が増加している。携帯電話上で低優先度通信 (Lower-than-Best-Effort) を実現することで、フォアグラウンドのアプリケーションに変更を加えることなく、フォアグラウンド通信の遅延を抑制することができる。本稿では、携帯電話上で低優先度通信を実現するために、低消費電力にパケットの優先制御を行う LP-LBE (Low-Power Lower-than-Best-Effort) の設計と実装、評価について述べる。LP-LBE では、実行時のオーバーヘッドが小さいソフトウェア割り込みハンドラをプライオリティキュー単位に割り当て、パケットの優先度に沿ってタスクとソフトウェアの割り込みハンドラの実行順序を制御する。実機上に LP-LBE を実装して評価を行った結果、消費電力の発生を抑制しつつ、フォアグラウンド通信の遅延を削減できることが確認された。

Low-Load Packet Scheduling for Lower-than-Best-Effort in Mobile Phones

TAKAHIRO YAMAMOTO,^{†1,†2} SHUNSUKE SARUWATARI^{†1}
and HIROYUKI MORIKAWA^{†1}

Background applications for mobile phones have been growing in recent years, such as feed reader, calendar application and mobile sensing application. When foreground application being operated by the user and background application transfer data at the same time, the throughput per session decreases and user operability is degraded. In this paper, we propose Low-Power Lower-than-Best-Effort (LP-LBE) protocol, which assigns the lower priority than best-effort to the background traffic and decrease the communication latency of the foreground application. LP-LBE reduces the energy consumption by processing packets in priority queues with software-interrupt handler and prevents priority inversions by scheduling both interrupt handlers and tasks based on the packet priority. Our estimation results show LP-LBE can reduce the communication latency of the foreground application to the same level with the previous work and save more energy than the previous work.

1. はじめに

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伴い、カレンダーアプリケーションや RSS リーダ、モバイルセンシング¹⁾ 等のユーザの操作を伴わずにバックグラウンドで定期的に通信を行うアプリケーションが増加している。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、ユーザが操作するフォアグラウンドの通信が遅延するという問題が生じる。

このようなフォアグラウンドの通信遅延の問題に対して、P2P やデータバックアップ等のバックグラウンドトラフィックが増大するインターネットの分野では、バックグラウンド通信の優先度を他の通信よりも下げてトラフィックを制御する低優先度通信 (Lower-than-Best-Effort)²⁾⁻⁷⁾ の研究が行われている。低優先度通信では図 1 に示すように、他の通信が実行されている間はバックグラウンド通信のスループットを下げ、実行されていない間はスループットを上げる動作によって、フォアグラウンドの通信遅延を抑制しつつ空いた帯域を有効活用する。バックグラウンドの通信に対して他の通信よりも低い優先度レベルを割り当てることで、ベストエフォートで通信を行うフォアグラウンドのアプリケーションには変更を加えずにトラフィックの優先制御を実現する。実際に P2P ツールである BitTorrent では、LEDBAT と呼ばれる低優先度通信プロトコルが採用されている^{6),7)}。

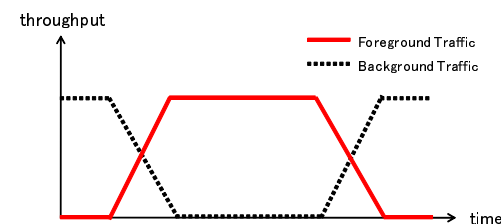


図 1 低優先度通信のスループット特性

^{†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

^{†2} 株式会社コア 総合研究所

CORE Research and Development Institute

低優先度通信を携帯電話上で実現することで、ユーザが操作するフォアグラウンドのアプリケーションに変更を加えることなく、フォアグラウンドの通信遅延を抑制することができる。携帯電話上で低優先度通信を実現するに当たっては、フォアグラウンドのアプリケーションの通信遅延が小さいこと、ソフトウェアによって実現可能であること、低消費電力であることの3つの条件を満たす必要がある。これまでに提案されている手法のうち、TCP層で低優先度通信を実現する方法^{4)–6)}では、フォアグラウンドトラヒックの有無をラウンドトリップタイムを使って検出することでトラヒックの制御を行う。そのため、バックグラウンドトラヒックの制御が開始されるまでの間は、フォアグラウンドの通信に遅延が生じる問題がある。IP層で低優先度通信を実現する方法では、ルータのキューを優先度別のプライオリティキュー⁸⁾に分け、ハードウェアによってパケットを優先度順に送出することでトラヒックの優先制御を行う。しかしながら、携帯電話ではソフトウェアによってトラヒックの優先制御を行う必要があるため、処理の実行順序とパケットの優先順位が一致しないパケットの優先度逆転⁹⁾が発生すると、フォアグラウンドの通信が遅延する問題が生じる。パケットの優先度逆転を抑制する方法については、プライオリティキューごとの処理にタスクを割り当て、タスクの優先度をパケットの優先度に沿って設定することで、パケットの優先度順にタスクを切り替える方法が提案されている^{9),10)}。しかしながら、プライオリティキューのパケットの処理をタスクで行った場合には、オーバーヘッドの大きいコンテキストスイッチ¹¹⁾が頻発するため、バッテリー駆動である携帯電話においてはCPU負荷の上昇に伴う消費電力の増大が問題となる。

このような携帯電話上で低優先度通信を実現する場合の課題に対し、本稿では、CPU負荷の上昇を抑制しつつパケットの優先度逆転を抑制する Low-Power Lower-than-Best-Effort (以下 LP-LBE) の設計と実装、評価について示す。LP-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、トラヒックの優先制御時の消費電力を削減するために、プライオリティキューに格納されたパケットの処理はオーバーヘッドの小さいソフトウェア割り込みハンドラによって行う。ソフトウェア割り込みハンドラを使用することによって発生するパケットの優先度逆転は、ソフトウェア割り込みハンドラとタスク優先度の関連づけ、パケット優先度ベースソフトウェア割り込みスケジューリング、パケット優先度ベースタスクスケジューリングの3つを行うことで抑制する。LP-LBEをAndroid端末上に実装して、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測定した結果、タスクを使用した既存手法と同等の遅延削減効果が得られることを確認した。また、トラヒッ

クの優先制御を行っている間のCPUの実行サイクル数を測定した結果、既存手法よりもCPUの実行サイクル数を18%削減できることを確認した。

本稿では、以下の構成でLP-LBEの設計と実装、評価について述べる。2では、低優先度通信の関連研究について述べ、既存手法を携帯電話に適用した場合の問題点を示す。3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LP-LBEの設計を示す。4では、Android端末上でのLP-LBEの実装方法について述べる。5では、LP-LBEを実装した端末を使用してフォアグラウンド通信の遅延時間とCPU負荷を評価した結果を示す。6では、まとめと今後の課題について述べる。

2. 関連研究

本稿では、携帯電話上で低優先度通信を実現することで、複数のアプリケーションが同時に通信を行ったときに生じるフォアグラウンド通信の遅延を抑制することを目指す。ただし、携帯電話では、IP層以上の通信プロトコルはソフトウェアで実現することと、容量の限られたバッテリーによって駆動することの2つの特徴を持つ。したがって、低優先度通信を実現する上では、フォアグラウンド通信の遅延削減効果が大いこと、ソフトウェアによって実現可能であること、低消費電力であることの3つの条件を満たす必要がある。

これまで低優先度通信に関する研究では、P2P等のバックグラウンドトラヒックが増大するインターネットの分野で進められてきており、TCP層によりEndノード間でトラヒックを制御する方法とIP層によりルータ上でトラヒックを制御する方法がある^{3)–6),8)}。TCP層

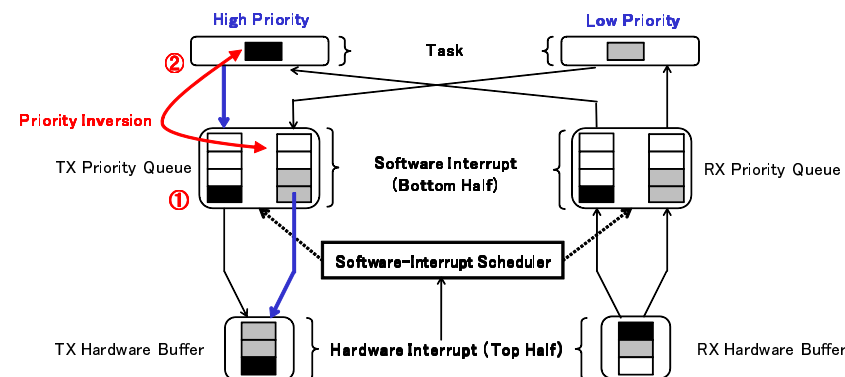


図2 パケットの優先度逆転問題

で低優先度通信を実現する方法としては TCP Nice⁴⁾ や TCP-LP⁵⁾, LEDBAT⁶⁾ 等がある。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、ルータを更新することなく低優先度通信を実現することができる。しかしながら、これらの研究では、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問題がある。

IP 層で低優先度通信を実現する方法としてはプライオリティキューイング (Priority Queueing)⁸⁾ がある。プライオリティキューイングでは、キューを優先度別に分け、優先度の高いキューの packets 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。ルータの場合にはプライオリティキューイングを並列処理可能なハードウェアによって実現するが、携帯電話の場合には逐次処理であるソフトウェアによって実現するため、実行順序の制御が必要となる。パケットの優先順位と処理の実行順序が一致しなければ、パケットの優先度逆転⁹⁾ が発生して、フォアグラウンドの通信に遅延が生じる。

図 2 に、トラヒックの優先制御をソフトウェアで実現した場合に生じるパケットの優先度逆転の様子を示す。ソフトウェアでは、逐次処理によってトラヒックの優先制御を実現するため、実行の制御単位となるタスクと割り込みハンドラの割り当てとスケジューリングが重要となる。1 つのタスクや割り込みハンドラの中で複数の優先度のパケットを処理すると、パケットの優先順に処理を切り替えることができずに、パケットの優先度逆転が発生する。また、割り込みハンドラとタスクのスケジューリングは独立して行われ、割り込みハンドラは常にタスクよりも優先して実行されるために¹²⁾、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割り込みハンドラの間でパケットの優先度逆転が発生する。

このようなパケットの優先度逆転を抑制する研究として、Tokuda らの研究⁹⁾ や Lim らの研究¹⁰⁾ がある。これらの研究では、プライオリティキューごとにタスクを割り当てることで、一つの処理の中で複数の優先度のパケットを扱うことで生じるパケットの優先度逆転を抑制している。また、ソフトウェア割り込みハンドラを使用せずに、タスクのみをパケットの優先度に沿ってスケジューリングすることで、タスクとソフトウェア割り込みハンドラの間で生じるパケットの優先度逆転を抑制している。しかしながら、プライオリティキューのパケットをタスクで処理することによって、コンテキストスイッチが頻発して CPU 負荷が上昇することから¹¹⁾、バッテリー駆動の携帯電話においては消費電力の増大が問題となる。

3. Low-Power Lower-than-Best-Effort

本節では、携帯電話上で低優先度通信を実現する Low-Power Lower-than-Best-Effort

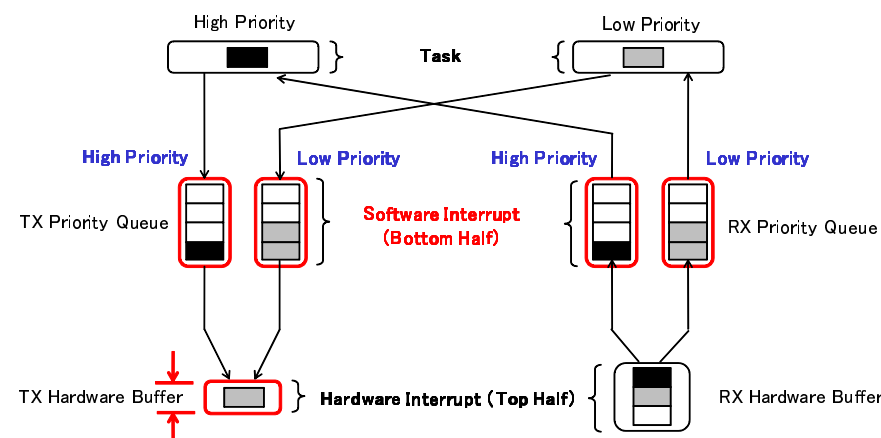


図 3 携帯電話向けプライオリティキューイング

(LP-LBE) の設計について述べる。携帯電話では、通信速度が低速であり、低消費電力性が求められることを考慮して、3.1 において携帯電話向けプライオリティキューイングの設計を行う。また、携帯電話では、IP 層以上の通信プロトコルが逐次処理であるソフトウェアによって実現されることを考慮して、3.2 においてパケットの優先度順に処理を実行するためのパケット優先度ベーススケジューリングの設計を行う。

3.1 携帯電話向けプライオリティキューイング

LP-LBE では、高優先度のパケット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、プライオリティキューを使用して IP 層でトラヒックの優先制御を行う。携帯電話では、無線アクセスリンクがバックボーンリンクに比べて狭帯域で、ボトルネックになることが多いため¹³⁾、端末から送出するパケットを IP 層で制御することで遅延時間を大幅に削減することができる。プライオリティキューイングを使った低優先度通信では、バックグラウンドのアプリケーションが接続先サーバの IP アドレスとポート番号を登録しておくことによって、フォアグラウンドで動作する既存のアプリケーションには変更を加えずにパケットの優先制御を実現する。

図 3 に、LP-LBE におけるプライオリティキューイングの構成を示す。プライオリティキューを使用するに当たっては、高優先度パケットの遅延時間がプライオリティキューとハードウェアバッファでの待ち時間の合計で表されることから¹⁴⁾、送信用ハードウェアバッ

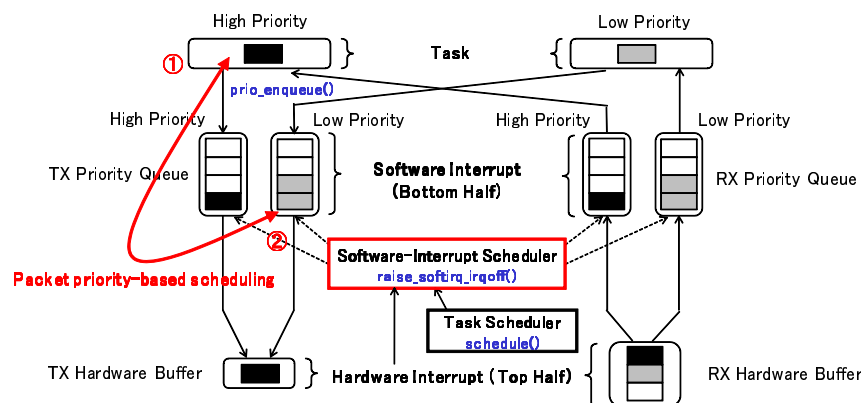


図 4 パケット優先度ベーススケジューリング

ファのサイズを 1 パケットの最大長まで縮小して低優先度パケットの挿入を抑制する。

IP 層でトラヒックの優先制御を実現する場合には、携帯電話では IP 層以上の通信プロトコルが逐次処理であるソフトウェアで構成されるために、パケットの優先度順に処理を実行する動作が必要となる。近年携帯電話では、機能の高度化に伴ってオペレーティングシステムを搭載しており¹⁵⁾、オペレーティングシステムの制御単位であるタスクと割り込みハンドラを割り当てて実行順序を制御することになる。割り込みハンドラには、ハードウェアが発生させる割り込みによって起動するハードウェア割り込みハンドラ (Top Half) と、ソフトウェアが発生させる割り込みによって起動するソフトウェア割り込みハンドラ (Bottom Half) がある。

LP-LBE では、消費電力の発生を抑制するために、オーバーヘッドの小さい割り込みハンドラを使ってトラヒックの優先制御を行う。ソフトウェア割り込みを発生させることによって、ソフトウェア割り込みハンドラを起動してプライオリティ・キューのパケットを処理する。実行頻度の高いパケットの処理にタスクを使用した場合には、処理を切り替える際にオーバー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^{11),16),17)}、IPC (Inter-Process Communication)^{18),19)} が頻発するために、CPU 負荷の上昇を招く。ソフトウェア割り込みハンドラを割り当てるに当たっては、一つの処理の中で複数の優先度のパケットを扱うと、優先度単位で処理を切り替えることができずにパケットの優先度逆転が発生するため、プライオリティキュー単位でソフトウェア割り込みハンドラを分割する。

3.2 パケット優先度ベーススケジューリング

携帯電話では、IP 層以上の通信プロトコルが逐次処理であるソフトウェアで実現されるため、実行順序の制御が重要となる。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り込みハンドラが独立したスケジューリング機構を持ち¹²⁾、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り込みハンドラがタスクよりも優先して実行される²⁰⁾。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り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割り込みハンドラの間でパケットの優先度逆転が発生する。

図 4 に、LP-LBE におけるパケット優先度ベーススケジューリングの様子を示す。LP-LBE では、タスクとソフトウェア割り込みハンドラの間の実行順序を制御することによって、パケットの優先度逆転を抑制する。具体的には、ソフトウェア割り込みハンドラとタスク優先度の関連づけ、パケット優先度ベースソフトウェア割り込みスケジューリング、パケット優先度ベースタスクスケジューリングの 3 つを行う。

まず、ソフトウェア割り込みハンドラとタスクの間で実行順序を制御するために、ソフトウェア割り込みハンドラとタスク優先度の関連づけを行う。LP-LBE では、タスクによってパケットがプライオリティキューに保存されたときに、プライオリティキューの優先度と

```

tid_current: task ID of the current task
prio_current: priority of the current task
prio_high: priority of foreground application task
prio_low: priority of background application task
tx_qlen: length of send queue
rx_qlen: length of receive queue

function raise_txsoftirq()
    if( (tx_qlen[HIGH]>0) && (prio_current >= prio_high || tid_current == 0) )
        raise TX.HIGH software-interrupt
    if( (tx_qlen[LOW]>0) && (prio_current >= prio_low || tid_current == 0) )
        raise TX.LOW software-interrupt

function raise_rxsoftirq()
    if( (rx_qlen[HIGH]>0) && (prio_current >= prio_high || tid_current == 0) )
        raise RX.HIGH software-interrupt
    if( (rx_qlen[LOW]>0) && (prio_current >= prio_low || tid_current == 0) )
        raise RX.LOW software-interrupt
    
```

図 5 パケット優先度ベースソフトウェア割り込みスケジューリング

実行したタスクの優先度を取得する。これら 2 つの優先度情報によって、プライオリティキューの packets を処理するソフトウェア割り込みハンドラとタスク優先度の関連づけを行う。タスク優先度を packets の優先度に沿って設定しておくことで、packets の優先度順にソフトウェア割り込みハンドラとタスクの実行を制御することができる。

次に、ソフトウェア割り込みハンドラの起動が packets の優先度に沿って行われるように、ソフトウェア割り込みスケジューラを変更する。図 5 に、変更したソフトウェア割り込みスケジューラを示す。優先度の値は小さいほど優先度が高いことを示す。LP-LBE では、ソフトウェア割り込みスケジューラが起動されると優先度の高いプライオリティキューから順に packets が存在するかチェックし、packets が存在した場合にはソフトウェア割り込みハンドラに関連づけられたタスク優先度と実行中のタスク優先度を比較する。実行中のタスクがソフトウェア割り込みハンドラよりも優先度が低いか、他に実行するタスクが無いときに実行されるアイドルタスクの場合には、ソフトウェア割り込みハンドラを起動する。packets が存在しないか、実行中のタスクがソフトウェア割り込みハンドラよりも優先度が高い場合には、ソフトウェア割り込みハンドラは起動せずに処理を終了する。実行中のタスクよりも優先度の高いソフトウェア割り込みハンドラのみを起動することで、高優先度 packets を処理するタスクと低優先度 packets を処理するソフトウェア割り込みハンドラの間で生じる packets の優先度逆転を抑制する。

最後に、タスクの起動が packets の優先度に沿って行われるように、タスクスケジューラを変更する。前述のソフトウェア割り込みスケジューラの変更を行うことで、実行中のタスクよりも優先度の低いソフトウェア割り込みハンドラは実行されずに、低優先度の packets がプライオリティキューに残る。LP-LBE では、実行中のタスク優先度が変化するタスクスケジューラの中でソフトウェア割り込みスケジューラを起動することで、プライオリティキューに残った低優先度 packets を処理する。次に実行するタスクの優先度が未実行のソフトウェア割り込みハンドラの優先度よりも低くなったときに、未実行のソフトウェア割り込みハンドラを起動することで、packets の優先度順に処理を実行する動作を実現する。

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 LP-LBE の実装を行った。

4.1 携帯電話向けプライオリティキューイング

IP 層を使ったトラフィックの優先制御を実現するために、携帯電話上にプライオリティ

キューを構築する。送信用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する。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 コマンドを実行することによって、送信キューを表す Qdisc 構造体を優先度ごとに作成する。送信用ハードウェアバッファのサイズは、高優先度 packets の遅延を削減するために縮小する。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 `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。 th の値は、packets サイズの上限である MTU の値 1500 byte を収容するために 1600 byte とした。受信用プライオリティキューは、受信 packets 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する。バックグラウンドのアプリケーションによって登録された IP アドレスとポート番号情報をもとに、バックグラウンド通信の packets を低優先度のプライオリティキューに格納することで、フォアグラウンドのアプリケーションに変更を加えることなくトラフィックの優先制御を実現する。

プライオリティキューに格納された packets 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り込みハンドラを使って処理する。Linux カーネルにおける送信用ソフトウェア割り込みハンドラである `net_tx_action` と受信用ソフトウェア割り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る。

4.2 パケット優先度ベーススケジューリング

ソフトウェアで packets の優先制御を行った場合に生じる packets の優先度逆転は、3 で挙げた 3 つの対策を実装することによって抑制する。具体的には、送信用プライオリティ

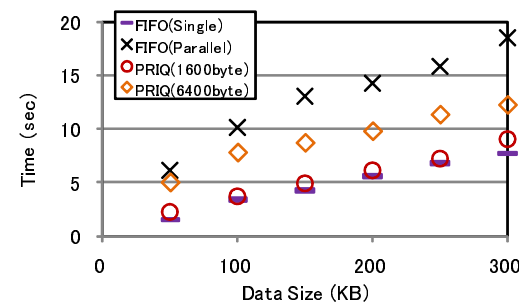


図 6 遅延時間の評価 (キューイング方式による比較)

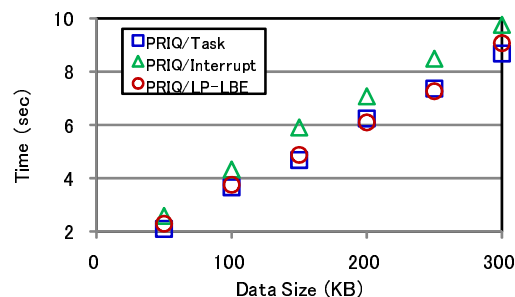


図 7 遅延時間の評価 (スケジューリング方式による比較)

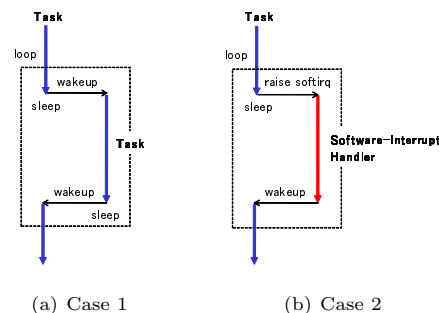


図 8 オーバヘッドの測定方法

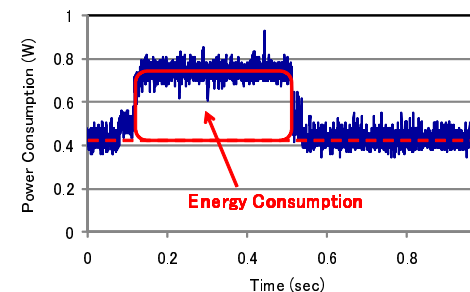


図 9 Case 2 を 30000 回実行したときの電力波形

キューにパケットを格納する `prio_enqueue()`、ソフトウェア割り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 3 つの関数を変更することによって、パケットの優先度順に処理を実行する。

ソフトウェア割り込みハンドラとタスク優先度の関連づけは、`prio_enqueue()` の中でパケットを格納したプライオリティキューの優先度と実行したタスクの優先度を取得することによって実現する。プライオリティキューを表す `Qdisc` 構造体と、実行中のタスク構造体を表す `current` の優先度 `static_prio` を取得することで、プライオリティキューのパケットを処理するソフトウェア割り込みハンドラとタスク優先度の関連づけを行う。パケット優先度ベースソフトウェア割り込みスケジューラは、`raise_softirq_irqoff()` の中で、実行中のタスクの優先度よりも優先度の高いソフトウェア割り込みハンドラのみを起動することで実現する。`raise_softirq_irqoff()` の中で、実行中のタスク優先度 `current->static_prio` と `prio_enqueue()` で取得したタスク優先度を比較し、`prio_enqueue()` で取得したタスク優先度の方が高ければ、ソフトウェア割り込みハンドラを起動する。パケット優先度ベースタスクスケジューラは、`schedule()` の中で `raise_softirq_irqoff()` を呼び出すことで実現する。次に実行するタスクの優先度 `next->static_prio` と `prio_enqueue()` で取得したタスク優先度を比較し、`prio_enqueue()` で取得したタスク優先度の方が高ければ、ソフトウェア割り込みハンドラを起動する。

5. 評価

本節では、LP-LBE を実装した Android 端末を使用して、フォアグラウンド通信の遅延

時間、パケットの処理を行ったときに生じるオーバーヘッド、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う。

5.1 遅延時間の評価

LP-LBE を開発用 Android 端末である GDD フォン上に実装し、WCDMA ネットワークを使用して、複数のアプリケーションが同時に通信を行った場合の遅延時間を測定した。バックグラウンドのアプリケーションが 500 KB のデータを送信している間に、フォアグラウンドのアプリケーションがデータ送信を行って、データサイズと通信時間の関係性を評価した。通信時間の値は 10 回の平均値を使用する。

図 6 に、キューイング方式の違いによる比較を示す。トラヒックの優先制御を行わない携帯電話の既定の状態ですべて通信を行った場合 (FIFO(Single))、並行して通信を行った場合 (FIFO(Parallel))、プライオリティキューイングによってトラヒックの優先制御が可能な状態で並行して通信を行った場合で比較した。プライオリティキューイングを使用する場合については、送信用ハードウェアバッファサイズの閾値 th を 6400 byte とした場合 (PRIQ(6400 byte)) と 1600 byte とした場合 (PRIQ(1600 byte)) で測定を行った。図より、携帯電話の既定の状態ですべて通信が行われると、フォアグラウンドの通信に遅延が生じて、単独で通信を行った場合に比べて通信時間が 2 倍になることが分かる。プライオリティキューイングを使用して低優先度通信を実現することにより、フォアグラウンドの通信遅延を抑制することができる。プライオリティキューイングを使用する場合には、送信用ハードウェアバッファのサイズを縮小することで、フォアグラウンドの通信遅延を削減することができる。

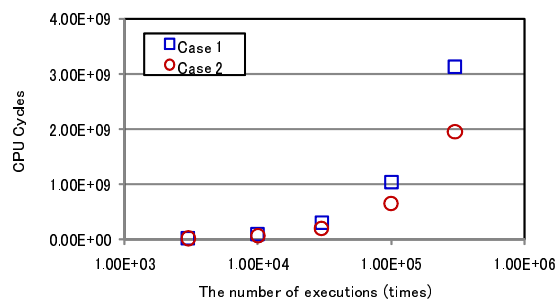


図 10 オーバヘッドの評価 (CPU 負荷)



図 11 オーバヘッドの評価 (消費電力)

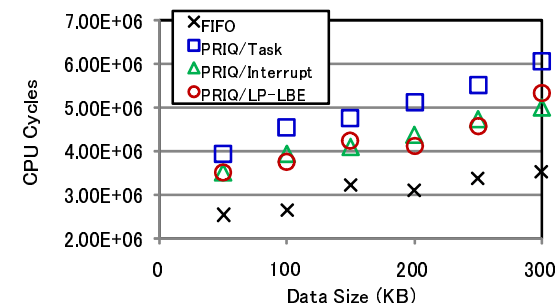


図 12 トラフィック制御時の CPU 負荷の評価

図 7 に、スケジューリング方式の違いによる比較を示す。プライオリティキューの packets の処理をタスクで行った場合 (PRIQ/Task), ソフトウェア割り込みハンドラで行った場合 (PRIQ/Interrupt), LP-LBE を使用してソフトウェア割り込みハンドラのスケジューリングを行った場合 (PRIQ/LP-LBE) で比較を行った。図より、タスクを使用した場合と LP-LBE を適用した場合とでは、パケットの優先度逆転を抑制することによって、フォアグラウンドアプリケーションの通信遅延を抑制していることが分かる。LP-LBE は、ソフトウェア割り込みハンドラとタスクの間の実行順序をタスク優先度に沿って制御することで、タスクを使用した場合と同等の遅延時間を実現することができる。

5.2 オーバヘッドの評価

次に、GDD フォンを使用して、ソフトウェア割り込みハンドラを使用した場合とタスクを使用した場合のオーバーヘッドの違いを測定した。図 8 に示すように、タスクとタスク (Case 1), タスクとソフトウェア割り込みハンドラ (Case 2) の切り替えのみを繰り返し行い、実行回数を変化させて CPU 負荷と消費電力量を評価した。CPU 負荷は ARM コアが持つ Performance Monitor の機能を使って実行サイクル数を取得することによって評価し、消費電力は端末とバッテリーの間に挿入した 0.1Ω のシャント抵抗の電圧をオシロスコープ DL750 で測定することによって評価した。図 9 に、タスクとソフトウェア割り込みハンドラの切り替え (Case 2) を 30000 回実行したときの電力波形を示す。消費電力量は電力波形の電力値を積算することで算出した。平常時の消費電力が 0.43 W と高いのは、プログラムの実行を指示する PC と通信するために USB コントローラを駆動していることによる。

図 10 に、切り替えの実行回数と CPU の実行サイクル数との関係を示す。実行サイクル数の値は 10 回の平均値を使用している。図から、ソフトウェア割り込みハンドラを使用す

ることによって、CPU の実行サイクル数を削減できることが分かる。測定結果から 1 回の起動当たりの実行サイクル数の違いを求めたところ、3944 であった。

図 11 に、切り替えの実行回数と消費電力量との関係を示す。消費電力量の値は 10 回の平均値を使用している。図から、ソフトウェア割り込みハンドラを使用することによって、消費電力を削減できることが分かる。測定結果から 1 回の起動当たりの消費電力量の違いを求めたところ、2576 nJ であった。

以上の結果から、タスクは起動する際のオーバーヘッドが大きく、多くの実行サイクルを必要とするために、ソフトウェア割り込みハンドラを使用した場合に比べて発生する消費電力が大きくなることが確認された。

5.3 トラフィック制御時の CPU 負荷の評価

最後に、トラフィックの優先制御を行っている間の CPU 負荷を測定した。バックグラウンドのアプリケーションが 500 KB のデータを送信しているときに、並行してフォアグラウンドのアプリケーションがデータ送信を行い、すべてのデータ転送が完了するまでに要した CPU の実行サイクル数を評価した。

図 12 に、プライオリティキューの packets の処理をタスクで行った場合 (PRIQ/Task), ソフトウェア割り込みハンドラで行った場合 (PRIQ/Interrupt), LP-LBE によってソフトウェア割り込みハンドラのスケジューリングを行った場合 (PRIQ/LP-LBE), 携帯電話の既定の状態で使用した場合 (FIFO) の比較を示す。実行サイクル数の値は 10 回の平均値を使用している。図より、LP-LBE を適用した場合は、タスクを使用した場合に比べて CPU の実行サイクル数を削減できていることが分かる。タスクを使用した場合には、パケットの処理を行うたびにコンテキストスイッチが発生するため、データサイズが多くなるにした

がってオーバヘッドの影響が拡大している。携帯電話の既定の状態では通信を行った場合は、トラヒックの優先制御を行わないために CPU の実行サイクル数が少なくなるが、通信遅延が発生するためユーザ操作に影響を及ぼす。1 KB 当たりの実行サイクル数の増加量は、タスクを使用した場合が 7933 であるのに対して LP-LBE を適用した場合が 6492 であり、LP-LBE を適用することで CPU の実行サイクル数を 18 %削減することができる。300 KB を送信したときのタスクを使用した場合と LP-LBE を適用した場合の実行サイクル数の違いは 746719 であり、5.2 の結果から 556.6 mJ の消費電力量に相当する。

6. む す び

本稿では、携帯電話において複数のアプリケーションが同時に通信を行う状況が増大していることを背景に、低消費電力に低優先度通信を実現する Low-Power Lower-than-Best-Effort (LP-LBE) について述べた。低優先度通信を携帯電話上で実現することで、ブラウザなどの既存のアプリケーションに変更を加えることなく、ユーザが操作するフォアグラウンドアプリケーションの通信遅延を抑制することができる。LP-LBE では、低消費電力性を実現するために、プライオリティキューごとのパケットの処理に実行時のオーバヘッドが小さいソフトウェア割り込みハンドラを使用する。また、パケットの優先度逆転を抑制するために、ソフトウェア割り込みハンドラとタスク優先度の関連づけ、パケット優先度ベースソフトウェア割り込みスケジューリング、パケット優先度ベースタスクスケジューリングの3つを行う。Android 端末上に LP-LBE を実装して評価した結果、既存手法と同等の通信時間を実現できることが確認された。また、パケットの優先制御を行っている間の CPU 負荷を既存手法よりも 18 %削減できることを確認した。今後は、無線 LAN などの通信条件やトラヒックパターンが異なる環境で測定を行い、LP-LBE による遅延削減効果と消費電力削減効果の評価を進める予定である。

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) R. Bless and K. Nichols and K. Wehrle: RFC3662: A Lower Effort Per-Domain Behavior (PDB) for Differentiated Services (2003).
- 3) Kokku, R., Bohra, A., Ganguly, S. and Venkataramani, A.: A Multipath Background Network Architecture, *IEEE INFOCOM*, pp.1352–1360 (2007).
- 4) Venkataramani, A., Kokku, R. and Dahlin, M.: TCP Nice: a mechanism for background transfers, *ACM SIGOPS*, Vol.36, pp.329–343 (2002).
- 5) Kuzmanovic, A. and Knightly, E.W.: TCP-LP: a distributed algorithm for low priority data transfer, *IEEE INFOCOM*, pp.1691–1701 (2003).
- 6) Rossi, D., Testa, C., Valenti, S. and Muscariello, L.: LEDBAT: The New BitTorrent Congestion Control Protocol, *IEEE ICCCN*, pp.1–6 (2010).
- 7) 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).
- 8) S. Blake and D. Black and M. Carlson and E. Davies and Z. Wang and W. Weiss: RFC2475: An Architecture for Differentiated Services (1998).
- 9) Tokuda, H., Mercer, C.W., Ishikawa, Y. and Marchok, T.E.: Priority inversions in real-time communication, *IEEE RTSS*, pp.348–359 (1989).
- 10) Lim, H., Park, D., Kang, S. and Oh, B.: Priority queue-based IEEE1394 device driver supporting real-time characteristics, *IEEE Transactions on Consumer Electronics*, Vol.46, No.3, pp.825–833 (2000).
- 11) David, F.M., Carlyle, J. and Campbell, R.H.: Context switch overheads for Linux on ARM platforms, *ACM ExpCS* (2007).
- 12) Zhang, Y. and West, R.: Process-Aware Interrupt Scheduling and Accounting, *IEEE RTSS*, pp.191–201 (2006).
- 13) Liedtke, J.: Pervasive data access in wireless and mobile computing environments, *Wireless Communications and Mobile Computing*, Vol.8, pp.25–44 (2008).
- 14) Ferrari, T., Pau, G. and Raffaelli, C.: Measurement based analysis of delay in priority queuing, *IEEE GLOBECOM*, Vol.3, pp.1834–1840 (2001).
- 15) Gartner: Forecast: Mobile Communications Devices by Open Operating System, 2007-2014. <http://www.gartner.com/it/page.jsp?id=1434613>.
- 16) Tan, T.K., Raghunathan, A. and Jha, N.K.: Energy macromodeling of embedded operating systems, *ACM TECS*, Vol.4, No.1 (2005).
- 17) Mogul, J.C. and Borg, A.: The effect of context switches on cache performance, *ACM SIGPLAN*, Vol.26, pp.75–84 (1991).
- 18) Liedtke, J.: On micro-kernel construction, *ACM SIGOPS*, Vol.29, No.5 (1995).
- 19) Parmer, G. and West, R.: Predictable Interrupt Management and Scheduling in the Composite Component-Based System, *IEEE RTSS*, pp.232–243 (2008).
- 20) Leyva-del-Foyo, L.E., Mejia-Alvarez, P. and de Niz, D.: Predictable interrupt management for real time kernels over conventional PC hardware, *IEEE RTSS*, pp.14–23 (2006).