

# Participatory Sensing における低消費電力転送エンジンの実現

## Implementation of Energy-Efficient Upload Engine for Participatory Sensing

山本享弘<sup>1,2</sup> 猿渡俊介<sup>1</sup> 森川博之<sup>1</sup>  
 Takahiro YAMAMOTO Shunsuke SARUWATARI Hiroyuki MORIKAWA

東京大学 先端科学技術研究センター<sup>1</sup>  
 RCAST, The University of Tokyo

株式会社コア 総合研究所<sup>2</sup>  
 CORE Research and Development Institute

### 1 はじめに

Participatory Sensing は、ユーザが所持する携帯電話を利用して、広範囲に渡る情報の収集を容易かつ低コストに実現する。携帯電話では消費電力が問題となることから、筆者らは低消費電力にセンサデータの転送を行う Piggyback Transport Protocol (PBTP) の検討を進めている [1]。本稿では Android 上での PBTP エンジンの実装を示す。

### 2 Piggyback Transport Protocol

携帯電話では、連続通信時の遅延軽減を目的として、データ通信完了後、送信データが無くても Inactivity Timer によって一定期間接続を維持する [2, 3]。すなわち、回線接続中であっても通信デバイスの空き時間が存在する。PBTP は、この空き時間を利用してセンサデータを転送することで、消費電力の低減を図るプロトコルである [1]。

PBTP エンジンを携帯電話上で実現するに当たっては、ユーザの通信の開始と終了を検出する必要がある。また、ユーザの通信とセンサデータの転送を並行して行うことにより、ユーザ操作に影響を及ぼすことが無いよう、ユーザの通信を優先処理する仕組みが必要となる。

### 3 PBTP エンジンの実装

PBTP エンジンを Android 上に実装する。PBTP エンジンは図 1 のように、Event Monitor, PBTP Controller, Priority Controller, PBTP Interface の 4 つのコンポーネントから構成される。PBTP エンジンは、2 で挙げた「ユーザの通信の優先処理」と「ユーザの通信の開始と終了の検出」を、パケットの優先制御と携帯電話上のイベント監視によって実現する。

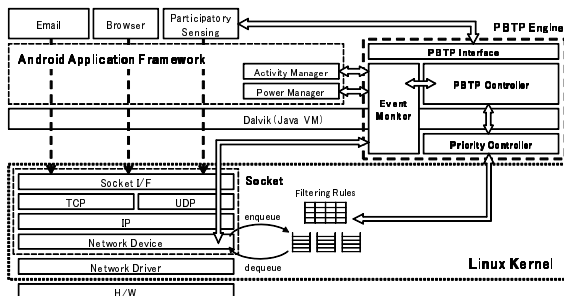


図 1 Android 端末上での実現方式

#### ● パケットの優先制御

PBTP エンジンでは、ユーザの通信の優先処理を、Priority Queueing (PRIQ) によるパケットの優先制御を行うことで実現する。Android 端末上では、(1)Linux カーネルへの QoS 機能の追加、(2)PBTP エンジンへの QoS 制御機能の追加、(3)PBTP エンジンによるスケジューリングポリシーの設定によって、パケットの優先制御を実現する。

Linux カーネルの既定の動作では、デバイスごとに割り当てられる送信キューは 1 つであり、パケットの送出手は FIFO によって行われる。Linux カーネルでは QoS 機能を追加することで、キューイング規則 (Queueing Discipline: qdisc) の変更が可能となる。そこで Android に含まれる Linux カーネルに対してコンフィギュレーションを行い、「QoS and/or fair queueing」を有効にして、QoS 機能を追加する。

Linux カーネルに対してキューイング規則の設定を行うために、PBTP エンジンに QoS を制御する機能を追加する。キューイング規則の設定は通常 tc コマンドによって行われるが、Android が提供するコマンドパッケージ toolbox では、tc コマンドをサポートしていない。そこで Priority Controller に tc コマンドの機能を実装する。

上記環境下で PBTP エンジンが、tc コマンドを使ってスケジューリングポリシーの設定を行うことで、パケットの優先制御を実現する。スケジューリングポリシーの設定は、キューの作成とフィルタリング規則の作成によって行う。キューの作成は、次のような tc qdisc コマンドを発行して行う。

```
tc qdisc add dev rmnet0 root handle 1: prio
スケジューリング方式として prio を指定することで、優先度の高い
```

キューから順にパケットの送信が行われる (図 2)。そのためパケットを異なるキューに振り分けることで、送信するパケットに優先順位を持たせることができる。パケットの格納先キューを決めるフィルタリング規則の作成は、次のような tc filter コマンドを発行して行う。

```
tc filter add dev rmnet0 parent 1:0 protocol ip prio 1
u32 match ip dst <IP address> flowid 1:1
```

フィルタリングは、IP パケットのヘッダ情報によって行われる。そのためアップロード先サーバの IP アドレスやポート番号を基に、センサデータを優先度の低いキューに格納するように設定することで、ユーザの通信を優先処理する動作を実現する。

#### ● 携帯電話上のイベント監視

PBTP エンジンでは、ユーザの通信の開始と終了を検出するために、Event Monitor が携帯電話上で発生するイベントを監視する。PBTP Controller は、ユーザの通信の開始を検出するとセンサデータの転送を開始し、ユーザの通信の終了を検出するとセンサデータの転送を停止することで、通信デバイスの空き時間を最大限活用する。

ユーザの通信の開始を検出するために、PBTP エンジンはユーザの通信によるパケットの発生を監視する。ユーザの通信によるパケットとセンサデータの転送によるパケットは異なるキューに格納されることから、キューからパケットの取り出しを行うネットワークデバイス層で検出することができる。ネットワークデバイス層が、ユーザの通信によるパケットの取り出しを検出し、シグナル等で Event Monitor に通知することにより、ユーザの通信の開始を認識することができる (図 2)。

ユーザの通信の終了を検出するために、PBTP エンジンはユーザによる通信アプリケーション操作の終了を監視する。Android プラットフォームでは、アプリケーションが利用する一連のプロセスをタスクという単位で管理する。そのため実行中状態にあるタスクの切り替わりによって、通信アプリケーションの操作が終了したことを判断する。アプリケーションを起動したまま放置したり、折りたたみ端末を閉じたりした場合は、タスクの切り替わりが発生しないため、スクリーンが消灯イベントによって判断する。実行中状態にあるタスク情報は、ActivityManager の getRunningTasks() によって取得する。スクリーンの消灯イベントである、PowerManager の ACTION\_SCREEN\_OFF は、BroadcastReceiver の onReceive() によって取得する。

```
for( prio=0; prio<q->bands; prio++){
  qdisc=q->queues[prio];
  skb=qdisc->dequeue(qdisc);
  if( skb ){
    sch->q.qlen--;
    if((prio<SENSOR_PRIO)&&(connection==OFF))
      signal(SIGUSR1, handler);
    return skb;
  }
}
```

図 2 ネットワークデバイス層における dequeue 処理

### 4 おわりに

本稿では、Participatory Sensing において、低消費電力にセンサデータの転送を行う PBTP エンジンの実装について述べた。今後 Android プラットフォーム上での PBTP エンジンの実装を進め、実機上での省電力効果の評価を行う予定である。

#### 謝辞

本研究は総務省からの委託研究「ユビキタスサービスプラットフォーム技術の研究開発」の成果である。

#### 参考文献

- [1] 山本ほか “Participatory sensing における低消費電力なセンサデータアップロードエンジン”, UBI2010-27, 2010.
- [2] J. H. Yeh, et al. “Comparative analysis of energy-saving techniques in 3gpp and 3gp2 systems”, Vehicular Technology, IEEE, 2009.
- [3] A. Larmo, et al. “The LTE link-layer design”, Communications Magazine, IEEE, 2009.