# Android 端末におけるカーネルモニタの導入

三木 香央理† 山口 実靖†† 小口 正人†

## 1. はじめに

近年, スマートフォン市場の成長に伴い, 携帯端末で動作する組込み機器のソフトウェアプラットホームとして Google 社開発の Android が注目されている. アプリケーション開発や柔軟な拡張性において注目度の高い Android 携帯に対し, 本研究ではそのネットワークコンピューティング能力について評価する. Android の様な組込み機器は, 汎用の PC などとはアーキテクチャが異なるため, インタフェースやリソースの問題から, 通信などの動作時に, 組込み機器の中でどのような事が起こっているのか, 正確に把握することは難しく, その通信動作を解析する事は興味深い. 本研究では, 組込み機器において, カーネルの中の振舞を把握することができるカーネルモニタツールを開発し,Android のトランスポート層においてこれを動作させて, 組込み機器の通信時の内部動作を解析することが可能である事を示す.

## 2. Android のアーキテクチャ

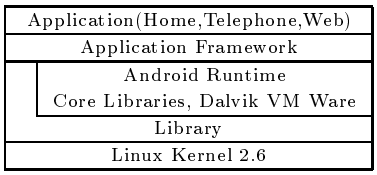


**図 1** Android のアーキテクチャ

Android のアーキテクチャを表 1 に示す. Android は Linux2.6 カーネルを用いて構築されており, この OS に各種コンポーネントを追加し Android というプラットフォームを構成している. また,Linux カーネルの上に Android 独自のアプリケーション実行環境である Android Runtime を実装し, Dalvik と呼ばれる独自の仮想マシンを搭載している. これは Java の仮想マシン（JVM）に相当する. その上にアプリケーション・フレームワーク, アプリケーションが乗る形態であるため, アプリケーションは Dalvik にあわせて開発すればよく, ポータビリティが高い.

† お茶の水女子大学
Ochanomizu University
†† 工学院大学
Kogakuin University

これまでの携帯端末用開発ツールとは異なり, オープンソースでありキャリア間の制約がないため, カスタマイズ自由度が高く, 他キャリアおよび他機種への柔軟な拡張性があるといえる.

一方, 通信については Linux カーネルの中のプロトコルスタックを用いて行われているため, この TCP 実装部分などで性能が決まってくると考えられる. そのため, 本研究ではカーネル中のトランスポート層実装に焦点を当て評価を行う.

### 2.1 カーネルモニタ

カーネルモニタは通信時に, どの時間にカーネルのコードのどの部分が実行されて, その結果カーネル内部のパラメータの値がどのように変化したかを記録する事ができるツールである.

図 2 に示すように, カーネル内部の TCP ソースにモニタ関数を挿入しカーネルを再コンパイルすることで TCP パラメータをモニタ可能にしている. これによりモニタできるようになった値には, 輻輳ウィンドウ, ソケットバッファのキュー長の他, 各種エラーイベント (Local device congestion, 重複 ACK, SACK 受信, タイムアウト検出) の発生タイミングなどがある. カーネルモニタよって, 正常動作時のカーネルの振舞を知る事ができ, さらには通信において問題が生じている場合に, その問題を特定し何が起こっているのか調べる事も可能となる.

本研究ではこのカーネルモニタを組み込み機器である Android に応用する.

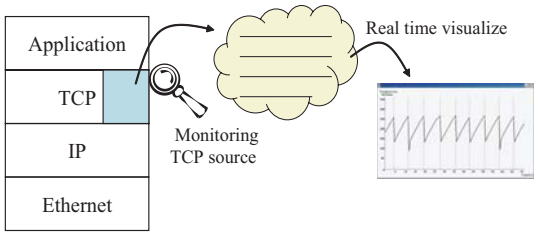


**図 2** Kernel Monitor

### 2.2 Android 端末のためのカーネルモニタ構築

Android は Linux カーネルをベースとしているが, 組込み機器であり汎用の PC と異なる点も多数ある. またシステムもアプリも開発はクロスコンパイルを用いる必要があり, OS のビルトは特殊な方法を用いて

行い, さらにコンパイルした OS を Android 携帯の実機で立ち上げるためにも特別な手順が必要となる.

本研究では, Android 携帯の実機向けの汎用 PC におけるカーネルモニタと同様のツールをを開発し, これをクロスコンパイルにより OS のコードに埋め込み, Android 携帯の実機に送り込んで立ち上げ, カーネルモニタが動作する事を確認した. そしてカーネルモニタを動作させて,Android の通信時の内部動作を解析することが可能である事を示す.

今回はその一例として,Android の通信における輻輳ウィンドウサイズとスループットの関係について解析する.

## 3. 実験システムと基本性能測定

本章では本実験で使用した測定ツール, 実験環境および実験手順を示す. 表 1 および図 3 に本研究の実験環境を示す. 本研究では. スループット測定を iperf-2.0.4[1] をクロスコンパイルし,Android-に送り込んでソケット通信の性能を測定した. クロスコンパイラとしては arm-2008q3[2] を使用した.

**表 1** 実験システム

| | | |
|---|---|---|
| Android | Model number | AOSP on Sapphire(US) |
| | Firmware version | 2.1-update1 |
| | Baseband version | 62.50S.20.17H_2.22.19.26I |
| | Kernel version | 2.6.29-00481-ga8089eb-dirty |
| | Build number | aosp_sapphire_us-eng 2.1-update1 ERE27 |
| server | OS | Fedora release 10 (Cambridge) |
| | CPU | CPU : Intel(R) Pentium(R) 4 CPU 3.00GHz |
| | Main Memory | 1GB |

### 3.1 Android 端末による遠隔サーバアクセスの通信性能

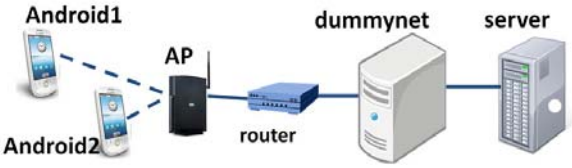


**図 3** 高遅延環境におけるサーバと 2 台の Android 間通信

図 3 に示すように, 間に人工的に遅延を発生させる装置である dummynet を用いて高遅延環境における 2 台の Android 端末とサーバ間の通信性能を測定した. これは遠隔地に存在するモバイルクラウドを提供するサーバへアクセスする通信を想定している.

### 3.2 2 台の Android 端末の通信における性能評価

図 4 は 2 台の Android 端末を 1 つの AP を使って通信させた際のスループットのグラフで横軸は時間である. このとき dummynet による遅延は発生させていない. 図 4 より 1 台の Android は比較的安定したスループットを得られているが, もう一台はスループットが大きく低下してしまい安定しない結果となることが確認された. 図 5 にカーネルモニタによって取得されたその時の輻輳ウィンドウサイズの変化を示す. スループットが安定している端末の方は輻輳ウィンドウサイズが低下してもすぐに増加しているのに対し, もう一台はなかなか輻輳ウィンドウサイズが増加できないことがわかる.RTT を大きくしていくとこの差はより明確に観測された.

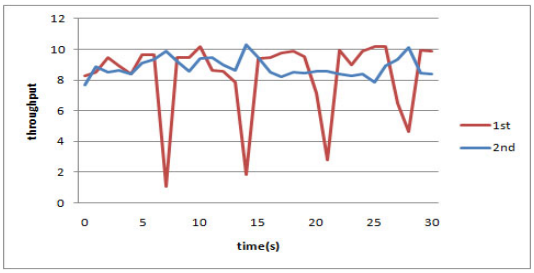


**図 4** 2 台の Android 端末と遠隔サーバの TCP 通信時スループット

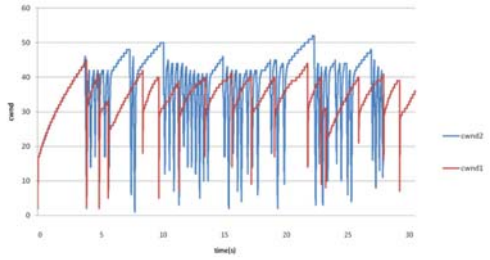

**図 5** 2 台の Android 端末と遠隔サーバの TCP 通信時輻輳ウィンドウ

## 4. まとめと今後の課題

今回は Android 実機にカーネルモニタツールを適応させ,Android 端末の通信時における輻輳ウィンドウの値を解析した.

今後はこのモニタツールを用いて輻輳ウィンドウサイズの効率的な増加方法や, 複数台の Android 端末で通信を行う際の効率的な通信方法を検討していきたい.

## 参 考 文 献

1) Iperf:http://downloads.sourceforge.net/project/iperf/iperf/2.0.4
2) Sourcery G++ Lite 2008q-3-72 for ARM GNU/Linux:http://www.codesourcery.com/, http://www.codesourcery.com/sgpp/lite/arm/portal/release644
3) Miki Kaori, Masato Oguchi, Saneyasu Yamaguchi, "A study about behavior of Transport layer on Android terminals in a wireless LAN," In Summer United Workshops on Parallel, Distributed and Cooperative Processing (SWoPP2010),August 2010.