Vol.2013-SLDM-160 No.30
Vol.2013-EMB-28 No.30
2013/3/14

Android における OS 可視化環境の開発

中川　裕貴[†1]　Praween Amontamavut[†1]　西野　洋介[†2]　早川　栄一[†3]

Android は，Linux カーネルと DalvikVM の二つの言語で構成されているので，複数のプロセスの動作を理解することが困難である．そこで，本報告では Android におけるプロセス可視化環境の開発を行った．本システムでは ftrace を用いた低オーバヘッドのシステム情報取得環境を構築し，プロセス生成や切換えに関する情報を取得可能とした．この情報を元に Web ブラウザで可視化環境を開発することで，ユーザが容易に利用可能な環境を構築した．本環境ではプロセスの OS モデル図，時間変化グラフ，プロセスのツリー構造を表示することにより，利用者がプロセスの実行時間や状態の遷移，プロセス同士の関係性を容易に把握できるようにした．

Development of a Visualization Enviroment for Android Operating System

YUUKI NAKAGAWA[†1]　Praween Amontamavut[†1]　YOUSUKE NISHINO[†2]
and EIICHI HAYAKAWA[†3]

Android is difficult to understand the behavior of processes, because it is built from Linux kernel and Dalvik VM. In this report we describe the development of the visualization about operating system process in Android. Data about context switch and process creation is available to acquire from ftrace based low overhead mechanism. Visualization environment is also developed that it executes on Web browser to be easily available to users. The environment consists of three components: state transition diagram of processes, time transition diagram and process tree diagram. User can understand the process execution time, the process state transition and the relationship among processes from these diagrams.

1. 背景

Android を携帯電話や組込み OS として利用する機会が多くなっている。その理由として、Android を搭載するスマートフォンを作る上で OS のライセンスの料金がかからないこと、開発者が無償で開発環境を入手でき、アプリケーション作成ができることが挙げられる。Android 利用者の増加に伴い、Android のアプリケーションの開発に必要となる OS の動作を理解する必要がでてきている。

しかし、AndroidOS の動作を理解するには多くの問題がある。一つには、ソースコードの規模が大きく、コードの動作を理解しにくい点である。Android はオープンソースであり、OS のコードレベルから調査をすることが可能である。しかし、ソースコードの規模は約 1600 万行と非常に大きく、全体の動作を把握することは困難である。また、Android 内では複数のプロセスが非同期に動作しているので、プロセスの動作をコードからトレースすることは難しい。

また、Android を組込みシステムで用いる場合、ハードウェアを扱うドライバをアプリケーションから利用する場合、デバイスドライバおよび JNI などのネイティブレイヤでのライブラリサポートが不可欠となる。一方、アプリケーション本体は Java で記述され、DalvikVM 上で動作している。ことから Android における動作をトレースする場合、ネイティブレイヤと DalvikVM レイヤの二つのレイヤを区別して扱う必要が出てくる。

開発者が OS の動作、特にプロセスやスレッドの動作をトレースしやすくするためには、その可視化環境としては、起動プロセスや CPU の可視化[1]が存在する。しかし、これまでに述べたように Android は Linux カーネルと DalvikVM の二つの階層で動作していることから、それらを関連付けて示すことが重要になっている。また、組込みシステムでは開発や学習環境のセットアップが複雑になりがちなことや、Windows などの OS に依存していることから、開発者にとって必ずしも使い勝手のよい環境になっていない。

2. 目的

本研究の目的は、可視化対象の OS を Android とし、階層化されたシステムにおける OS 資源のロギングおよび可視化を行うことである。特に、プロセスやスレッドに着目し、ロギング、分析、可視化を行う環境を作成した。また、可視化環境をブラウザで実行することにより、実行結果のアプリケーションのインストールをする必要がなくなる。また、ネット環境があれば使用できるので、学習者が利用しやすくなる。

本研究の特徴は、可視化の実行画面をブラウザ上で表示

†1 拓殖大学　大学院　電子情報工学専攻
Graduate School of Engineering, Electronics and Information Science, Takushoku University
†2 東京都立　八王子桑志高等学校
Hachiouji Sousi High School
†3 拓殖大学　工学部　情報工学科
Faculty of Engineering, Department of Computer Science, Takushoku University

することである。Web ブラウザで表示することで、アプリケーションのインストールが不要になることで、教育目的で使用する場合などで大量の PC にインストールする必要がなくなる。また、ネット環境があれば使用できるので、学習者が利用しやすいという利点がある。

可視化の対象は Android とする。携帯端末やホームアプライアンスなどの多くの環境で利用されているからである。

3. 問題分析

Android は携帯端末を主なターゲットとして開発されたプラットフォームである。多くのアプリケーションもあり、開発環境やドキュメントなども充実している。しかし、Android の OS 学習については勉強会やテキストしかなく、アプリケーションのようにネットなどで調べながら簡単に開発できるものが少ない。特にアプリケーションを動作させながら、呼び出している OS の機能の理解や、演習として OS の機能の改良や拡張を行う場合、OS の構造や動作がイメージしにくく、開発者や学習者にとって理解が困難である。

また、Android は C 言語と Java 言語の二つの階層で動作していることから、それらを関連付けて示すことが重要になっている。Android は携帯端末用に開発されているのでプロセスが従来の OS との違い理解することが難しい。Android は Linux カーネルの C 言語と DalvikVM の Java 言語の二つの階層で動作している。この C 言語と Java 言語の二つの階層でプロセスはどのように生成されているのか理解が難しい。そして、プロセスが C 言語と Java 言語の二つの階層でどのような動きをしているのかがつかみにくい。なので、Android の OS 全体の動きが分かりにくいという問題がある。

4. 設計

4.1 Version1 のシステム概要

これらの問題を解決するため、拓殖大学工学部情報工学科早川研究室では、Android のプロセスを可視化するツールを開発した(以下 Version1 と呼ぶ)。Version1 では可視化対象を Android として、プロセス管理の可視化[2]が可能になった。このツールの開発を通して、次の問題点が明らかになった。

- 可視化対象のプロセスが多くなると可視化結果が見にくくなる。
- 可視化するプロセスが多い時と少ない時のアニメーションが同じ場合、可視化結果の見落としが生じる。
- ツールの拡張性が不十分であり、他の可視化ツールを統合することができない。

4.2 設計方針

可視化ツール Version1 の問題点を解決し、新たに Version2 を開発した。Version1 の問題に対する改良点は次の通りである。

(1) 可視化の表示をコンポーネント化する

ログデータをもとに可視化の表示を行っているが、利用者が知りたい情報が表示されていない場合があるので、可視化情報を変更できるようにした。可視化の表示部分をコンポーネント化されており、可視化表示するプログラムが分割されている。分割された可視化表示部分のプログラムを変えることで様々な可視化の表示が可能となり、利用者が知りたいプロセスの可視化情報を表示することが可能になっている。

(2) 可視化表示するプロセスを選択可能にする

可視化表示する際にプロセスが多いと可視化したプロセス情報が見にくいという問題があったので、プロセスの可視化表示を行う前に可視化するプロセスを選択できるようにした。可視化で表示したいプロセスを選択可能になったことで、利用者が知りたいプロセスの情報を表示できるようになった。

(3) アニメーションの表示速度を調整可能にする

アニメーションの表示をする際に早すぎるや遅すぎるといった意見があったので、アニメーションの表示速度の調整が可能にすることで解決した。アニメーション速度を利用者の好みで変更可能になったことで、好みの速さで可視化のアニメーションを表示することで利用者に実行結果が見やすくなる。

4.3 全体構成

Androidにおけるプロセス可視化環境の全体構成を図1に示す。Android内にあるLinux内にあるトレーサーのftraceとAndroid内にあるデバック用ツールであるlogcatのログデータをサーバへ転送し、ログファイル生成する。データサイズが大きくなるので、ログの保存をAndroid内でなく、サーバ内に保存する。そのようにすることにより、Android内の保存領域を使用せずにすむ。サーバ内で可視化するためにログファイルを加工する。

Android 内で保存する場合、大きなサイズのログファイルの保存領域を取ることは難しい。しかし、Android 内にログファイルを保存できる保存領域を持っていない場合でも、サーバに保存することによりログファイルを生成し、保存することが可能である。

Vol.2013-SLDM-160 No.30
Vol.2013-EMB-28 No.30
2013/3/14

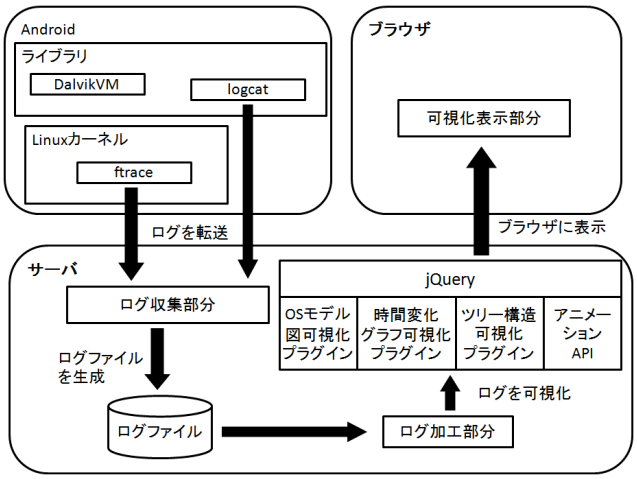

図 1　全体構成

加工したログファイルを JavaScript で可視化し、表示する。可視化のアニメーションには jQuery という HTML と JavaScript の相互作用を強化する軽量な JavaScript ライブラリを使用する。jQuery は既存のプラグインを使用すること以外にも新しくプラグインを作成することができる。本システムでは、三つのプラグインを作成した。状態遷移の可視化を表示する際に使用する OS モデル図可視化プラグイン、時間変化グラフの可視化を表示する際に使用する時間変化グラフ可視化プラグイン、ツリー構造の可視化を表示する際に使用するツリー構造可視化プラグインの三種類を作成した。また、アニメーション API を作成した。アニメーション API は可視化プラグインで描画したものをアニメーションで表示するときに呼び出される API である。このアニメーション API は、三つの可視化表現以外に表示したい場合に可視化プラグインを変えるだけで新しい可視化の表示することが可能である。

可視化の実行画面をアニメーションで表示する。アニメーションで表示することにより、利用者に可視化の実行結果が見えやすくなる。動的に表現することにより利用者が視覚的に理解できる。

可視化するプロセスは表示したいものだけを選択することが可能である。可視化で表示したいプロセスを選択することにより、利用者の知りたいプロセスの情報だけを見るとこが可能になっている。

4.4 プロセス情報取得

Android の可視化を行うためにプロセスの状態遷移や次に実行されるプロセスといった情報を取得する必要がある。プロセスの情報取得には Android の Linux カーネル内にある ftrace と Android のデバック用ツールの logcat を利用することでプロセスの情報の取得を行う。この機能を利用することで、プロセスの生成やコンテキストスイッチの様子をトレースすることができる。プログラムを実行させ、そのプログラムが実行している間のプロセスをトレースして状態遷移などの情報も取得する。プロセスの状態とそれに準ずるプロセスの色を表 1 に示す。

表 1　プロセスの状態と色

ftraceでの表記	状態	意味	色
R	Running	実行状態もしくは実行可能状態	緑もしくは黄色
S	Sleep	待ち状態（入出力とは無関係に遷移する状態）	赤
D	Disk sleep	ディスクへの読み書きによる待ち状態	青
T	Stopped	停止状態	ピンク
t	traced	トレース中	オレンジ
Z	zombie	ゾンビ状態	紫
X	unknown	存在しない	白

logcat で使用するデータはプロセス ID だけとなっている。logcat のプロセス ID と ftrace のプロセス ID が同じものは Java 言語と C 言語の二つの言語環境間で動作している。二つの言語間で動作しているプロセスは可視化の表示方法を変えている。表示方法を変えることにより、プロセスがどのように二つの言語間で対応付けられているのかを知ることができる。

また、学習者は実行させるプログラムの内容に関係なく、そのプログラムが Android 内部でどのようにプロセスとして生成され状態遷移をすることで実行されているのか理解することができる。Linux カーネルと DalvikVM の二つの階層で動作しているプロセスを知ることもできる。

4.5 可視化手法

Android のプロセス可視化部分は次の三つで可視化をする。[3]

(1) OS モデル図

プロセスの状態を 3 種類に分けて表示していく。3 つの状態は実行状態、実行可能状態、待ち状態となる。DalvikVM と Linux カーネル上の二つの言語間で動作しているプロセスはプロセス名の文字と色を薄くして表示している。時間変化グラフとツリー構造も同様に色を薄くして表示している。この可視化を表示することで、消滅できずにゾンビ状態のままでいるプロセスを見つけることができる。

(2) 時間変化グラフ

各プロセスの状態遷移を棒グラフによる表示をする。プロセスの実行状態、実行可能状態、待ち状態は三つの色に分けてグラフ上に表示する。Linux では、待ち状態には 6 状態があり、それらは色を変えて表示している。入出力とは無関係に遷移する待ち状態、ディスクの読み書きの待ち状態、停止状態のための待ち状態、トレース中による待ち状

Vol.2013-SLDM-160 No.30
Vol.2013-EMB-28 No.30
2013/3/14

態、ゾンビ状態による待ち状態、存在しないための待ち状態といった 6 種類の待ち状態を可視化していく。この可視化を表示することで、プロセスのデッドロックが分かりやすくなる。また、状態が遷移したタイミングの正確な時間が認識できるようになっている。

(3) ツリー構造

プロセス同士の繋がりを目で見えるようにするために可視化する。プロセスの親子関係や他のプロセスとの関係性をツリー構造で表示していく。また、プロセスの生成や消滅を表示することにより、いつプロセスが生成または消滅するかを知ることができる。この可視化画面を表示することで親プロセスが消滅したが、子プロセスが消滅していないプロセスを見つけることが可能である。

4.6 可視化画面

可視化画面を Web ブラウザ上で表示させており、JavaScript で書いている。そして、JavaScript のプラットフォームである jQuery を用いて可視化を表示している。jQuery は既存のプラグインを使用すること以外にも自分で新しいプラグインを作成することができる。今回、三つのプラグインを作成した。それぞれを OS モデル図可視化プラグイン、時間変化グラフ可視化プラグイン、プロセスツリー可視化プラグインとして可視化の表示を行う際に使用した。また、本ツールでは可視化の表示をした後にアニメーションでプロセスの状態遷移を表現している。その際には、アニメーション API を使用することでアニメーションの表示を効率的に行うことが可能になった。

ログデータの取得には、JSON 形式を利用している。JSON 形式で ftrace のログデータと logcat のログデータの取得を行う。プロセスの可視化表示には ftrace のログデータを利用する。logcat のログデータは DalvikVM と Linux カーネルの二つの階層で動作しているプロセスの情報の取得を行うために使用している。どのようにプロセスが二つの階層で動作しているか調べる方法として、ftrace のログデータ内のプロセス ID と logcat のログデータ内のプロセス ID が同じプロセスが 2 つの階層で動作しているプロセスとなる。そのプロセスは他のプロセスとは色の配色を変えることで表現をしている。次の図 2 で実行画面でのプロセスの変化を表す。

図 2 では、赤枠のプロセスが Linux カーネルと DalvikVM の二つの階層で動作しているプロセスである。このようにプロセスの名前の文字とプロセスの状態の配色を変えることで表示を行っている。このように表示することで他にプロセスとの違いが分かりやすいようになっている。

図 2 の実行画面の左上に時間変化グラフが表示されており、時間変化によるプロセスの状態の遷移を表示している。左下は OS モデル図が表示せれており、プロセスの状態の変化を表示している。右下にはツリー構造が表示されており、プロセス同士の関係性を表示している。

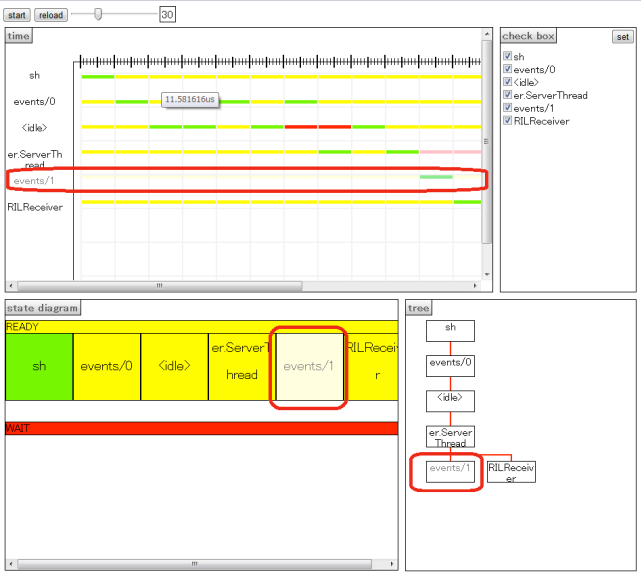

図 2　実行画面でのプロセスの変化

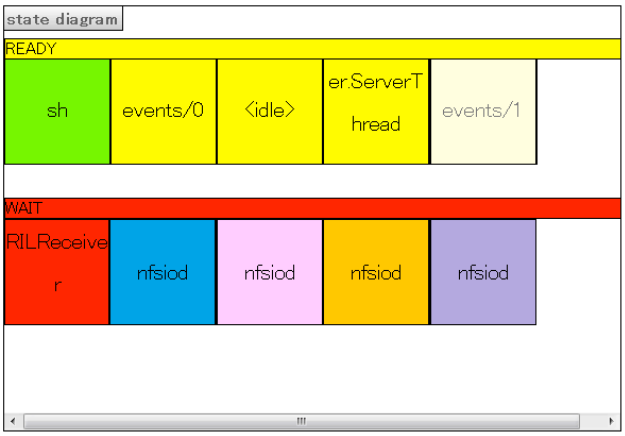

図 3　OS モデル図

図 3 が OS モデル図を可視化したものである。四角形のボックスの中にはプロセス名があり、その上にある横に長い長方形はプロセスの状態について表示している。プロセスの状態が遷移する度にボックスが次の状態へと移動し、ボックスの色も次の状態の色に変化する。

また、プロセスの待ち状態では 6 種類の状態を表現している。それらが、上図の下の段に表示されている。五つのプロセスしか存在しないが、存在しない状態という状態が消滅状態なのでこの図の中では消えてしますので、確認できない。それ以外の状態は左から、待ち状態、ディスクの読み書きによる待ち状態、停止状態、トレース中による待ち状態、ゾンビ状態という順番で表示されている。

Vol.2013-SLDM-160 No.30
Vol.2013-EMB-28 No.30
2013/3/14

図 4　時間変化グラフ

図 4 が時間変化グラフを可視化したものである。縦軸にプロセス名､横軸に状態遷移のタイミングを表示している。プロセスの状態を色で表現しており、状態の色は OS モデル図の配色と同じである。時間の経過に伴い、グラフが左に移動していく。

利用者がプロセスの状態遷移のタイミングを正確な時間で知りたい場合、グラフ上で知りたい部分にマウスカーソルを合わせるとその知りたい部分の実行時間がポップアップで表示される。これにより、自身の知りたいプロセスの実行時間を知ることが可能になった。

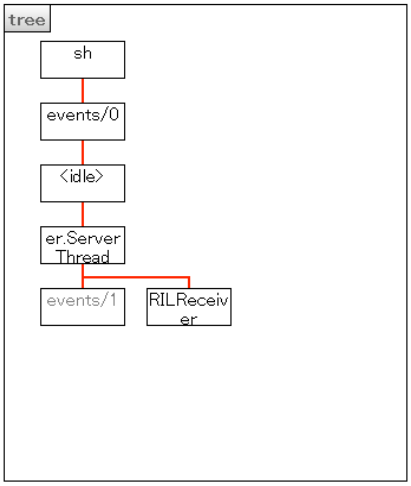

図 5　ツリー構造

図 5 がツリー構造を可視化したものを表示している。四角の中にはプロセス名があり、赤い線の繋がりで親子関係を表している。赤い線で繋がっている上のプロセスが親プロセスで下に繋がっているプロセスが子プロセスである。また、この可視化画面はプロセスの生成、消滅をプロセスの表示と非表示のアニメーションで表示している。

5. 実装

5.1 実装環境

環境は Eclipse の ver4.2　Juno を使用し、OS は Android ver2.3　Gingerbread を使用し、実機は KZM - A9 - Dual を使用した。

5.2 可視化表示部分の実装

可視化のプロセスを表示、非表示にすることができ、表示したいプロセスをチェックボックスで選択できるようになっている。プロセスの一覧が縦一列に並んでおり、名前の隣にチェックボックスがある。このプロセスの一覧はログデータに表示されているプロセスが表示されている。チェックしたプロセスだけを表示している。プロセス選択画面を図 6 に示す。

図 6　プロセス選択画面

選択したプロセスのみ可視化の表示が行われる。選択したプロセスのみを表示した実行画面を図 7 に示す。

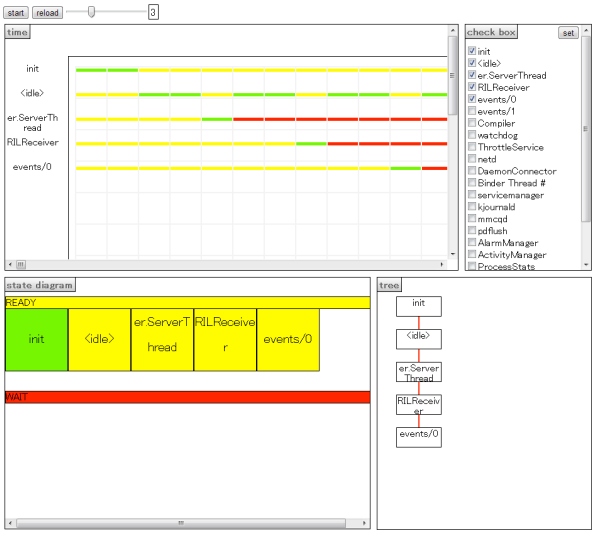

図 7　プロセス選択後の実行画面

上図はプロセスが 20 あり、その中で 5 つのプロセスを選択し可視化表示を行っている。可視化するプロセスを選択することで、利用者の知りたいプロセス情報を表示することが可能である。これにより、多くのプロセス情報の中から知りたい情報のみを表示することが可能になった。

可視化の表示には、アニメーションを使用している。アニメーションは表示速度を調整することが可能である。表示速度の調整方法は、スライドバーで表示速度の調整ができるようになっている。アニメーション表示速度が 1～10 秒の間で調整ができるようになっている。標準の設定では

3 秒となっている。可視化のアニメーションを表示する前に調整することで可視化のアニメーションの表示速度の調整が可能になっている。

5.3 可視化画面

Android におけるプロセスの可視化をすることにより、周期的に実行すべきプロセスのデッドラインが近いかを知ることができる。また、実際に Android を動かしログデータの取得し、可視化を行った。その結果、様々なプロセスの状態を取得することができ、AndroidOS 上でのプロセスのバグを見つけることが可能となる。その実行画面を図 8 に示す。

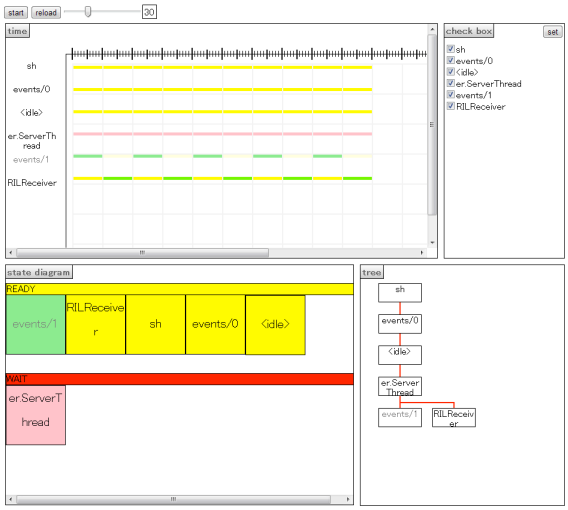

図 8　実行画面

この図 8 の実行画面は er.ServerThread というプロセスが停止状態の時の画面である。このプログラムは二つのプロセスがお互いに実行状態に移り変わり、er.ServerThread が実行可能状態に遷移できないバグである。このように AndroidOS 上のプロセスのバグを見つけることが可能となっている。

6. 関連研究

関連研究として、次の研究について述べる。

- TraceLogVisualizer[4]

マルチプロセッサの可視化を行った。トレースログを一般的に扱えるように標準形式を定める。任意の形式のトレースログを表示形式に変換するための変換ルールを定めた。また、トレースログの可視化表示の表現方法を指示する仕組みを抽象化した。これら可視化ルールはユーザから指定できる仕組みを提供している。

一方、この可視化ツールには可視化の表示を行うまでに時間を要することである。この可視化ツールは変換による表示形式変換や可視化ルールによる図形のデータを生成しないと可視化表示を行えないので可視化に要する時間が長い問題がある。本研究では、可視化ツール上でデータの生成を行わないので、サーバ内のログデータと通信するだけなので可視化に要する時間は少ない。

- Android 端末における通信性能の可視化ツール[5]

このことを解決するために Android 端末の通信性能の可視化を行った。この可視化ツールの特徴として、モバイル端末上での可視化であることが挙げられる。スマートフォンは移動する端末であるため、端末のみで測定や解析が行えるので有効である。このツールを用いて、通信時の輻輳ウィンドウサイズとその時のスループットの可視化を行った。輻輳ウィンドウサイズとは、受信側からの確認応答無しで連続送信できる最大パケット数を表す。

しかし、この可視化ツールには、複数の Android 端末が一つのアクセスポイントに接続した場合、通信状況を他の Android 端末が知ることができないという問題がある。他の Android 端末の通信状況を知ることで通信のスループットの違いを見ることができる。本研究では、同一の可視化結果を複数の PC で見ることできるので、他の実行結果と比較することも可能である。

7. おわりに

本研究では、可視化対象の OS を Android とし、プロセス管理の可視化を行った。プロセスの状態遷移の様子を取得する機構を実現したことにより、複数あるプロセスの動作を理解することが可能になった。Android の可視化結果を Web ブラウザ上で表示を行った。

今後の課題としては、Java 言語で動作している DalvikVM の可視化の表示を行えるようにすることである。また、可視化の実行画面のプロセスの選択画面、画面構成の変更などの改良を行ったが、改善の余地があるので改良を行っていくことである。

謝辞　本研究の一部は科学研究費補助金（基盤研究(C) 22500936）の助成を受けて実施した.

参考文献

1)　安藤 友樹、柴田 誠也、本田 晋也、冨山 宏之、高田 広章：組込みマルチプロセッサシステムの設計改善支援、SWEST12（2010）pp.54-56
2)　中川 裕貴、早川 栄一、西野 洋介：Android における OS プロセス可視化環境の開発、情報処理学会研究報告. EMB, 組込みシステム(2011)pp.1-6
3)　本橋 大樹、西野 洋介、早川 栄一：組込みシステム学習支援環境「港」における Linux プロセス可視化環境の開発（コンピュータシステム）、電子情報通信学会（2010）pp.279-285
4) 後藤　準弐、本田　晋也、長尾　卓哉、高田　広章：トレースログ可視化ツールの開発、情報処理学会研究報告. EMB(2009)pp.73-78
5) 平井　弘実、三木　香央理、山口　実靖、小口　正人：Android 端末における通信性能の可視化ツール、情報処理学会全国大会(2011)pp.213-214