Android端末の様々な無線LAN環境における通信性能の考察

三木香央理† 小口 正人†

† お茶の水女子大学 〒112-8610 東京都文京区大塚 2-1-1
E-mail: †kaori@ogl.is.ocha.ac.jp, ††oguchi@computer.org

あらまし 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, 本研究ではそのサービス提供を可能にしたシステムプラットホームとしての Android に焦点を当て, 特にそのネットワーク能力および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 携帯の無線ネットワークにおける通信能力について解析し, そのトランスポート層の仕様に手を加えることで, より高性能な通信を目指す.
キーワード Android, スマートフォン, 組込み機器, モバイル端末, ソフトウェアプラットフォーム

A study about performance of communication on Android terminals in a wireless LAN environment

Kaiori MIKI† and Masato OGUCHI†

† Ochanomizu University 2-1-1 Otsuka, Bunkyouku Tokyo 112-8610 JAPAN
E-mail: †kaori@ogl.is.ocha.ac.jp, ††oguchi@computer.org

Abstract In reccent years, with the rapid growth of smart phone market, Android is drawing an attention as software platform of embedded system on personal digital assistant developed by Google. While Android is taken notice for flexible development of application software and expansion of the system, we are interested in Android as a system platform, and in particular, we aim to optimize and evaluate performance of network computing ability. We have evaluated detailed behavior of Android machines in the wireless LAN environment, and we have modified mechanism in Transport layer and we aim to achieve higher performance of network communications.
Key words Android,smart phone,Embedded system,Personal Digital Assistant,software platform

1. はじめに

近年, １人１台の携帯電話を所有することが当たり前となってきている. また１人で複数台所有することも稀ではなく, サービスや用途によって使い分けているユーザが増加している. 以前は音声通話とメールが主な使われ方であったが, 最近は通信速度が向上し, インターネットや音楽, 動画, ラジオ, テレビ, 非接触型 IC など多機能化されている. 従って, キャリアごとに仕様が違う OS を用いると開発に膨大なコストが掛かるため, 独自の OS では対応しきれなくなっている.

そこで求められてきたのが携帯電話向けの汎用 OS である. 基本部分はプラットフォームとして共有化し, 独自の機能やサービスは個別に開発する. これによって開発の効率化が実現できる. またプラットフォームを公開し, オープンソフトウェアにすることで, 対応アプリケーションが作りやすくなり数も増えるというメリットがある. これを実現したものが Google 社開発の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,Android [1] である.

Android は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るためアプリケーション開発における制約がない. また Android 搭載の携帯上でならキャリア, 端末を問わずアプリケーションを実行でき, カスタマイズの自由度が高い. これらの要因から多くのユーザ間でシェアが広まってきている. この様に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, 本研究ではそのサービスを提供することを可能にしたシステムプラットホームとしての Android に焦点を当て, 特にそのネットワーク能力およびネットワークコンピューティング能力について掘り下げていく. 本論文では Android 携帯の無線 LAN の通信能力について解析し, そのトランスポート層の仕様に手を加えることで, より高性能な通信を目指す.

2. 研究背景

2.1 Android

Android のアーキテクチャを図 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.Linux のうち必要最低限の動作を担うカーネル部分だけが採用されているため, 市場に複数ある Linux パッケージでも Android が構成できるように設計されている.

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.

このように, 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. そのためユーザのカスタマイズ自由度が高く, アプリケーション開発の負荷が軽減され, 他キャリア, 他機種への柔軟な拡張性があるといえる.

一方, 通信は 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.

図 1 Android のアーキテクチャ

2.2 クロス開発

携帯端末はディスプレイも小さく,CPU 性能やメモリ容量が高くないため, 開発時と実行時に異なるコンピュータ環境を用いるクロス開発の形が取られることが一般的である. 主に開発を行うコンピュータをホスト環境といい,Android 実機をターゲット環境という. ホスト環境に通常無いカメラ機器等はホスト内にある模倣ソフトウェアであるエミュレータを動かすことで実行できるようになっている. また,Android は組込み機器であるため, 実機で実行できるコマンドにも制限があることから開発環境としては適していない. クロス開発を行うことで開発の効率を上げることができる.

3. 実験システム

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. 今回は 3 パターンの無線 LAN 通信のケースを想定しその性能を評価した. 表 1〜2 および図 2〜4 に本研究の実験環境を示す.

3.1 実験環境

スループット測定は iperf-2.0.4 [2] をクロスコンパイルし,Android 実機に送り込んで実行した. クロスコンパイラとしては arm-2008q3 [3] を使用した. また Android 実機に搭載されているコマンド数が少ないため, 様々なコマンドのアプレットに対応した busybox-1.10.1 [4] をインストールした.

3.2 測定環境

3.2.1 Wi-Fi 環境における同一無線 LAN 内の通信

図 2 Wi-Fi を用いた AP 経由の通信

まず,Wi-Fi 無線 LAN 機能を用いて AP を経由した場合の通信性能を測定した. これは図 2 に示すように AP の近くにいるユーザ同士の通信を想定している. 2 台の Android 端末と 1 台の Linux 端末（Ubuntu）の間で通信を行いスループットを測定した.

3.2.2 Wi-Fi のない環境におけるアドホック通信

図 3 Bluetooth を用いた AP を経由しない通信

次に,Wi-Fi を用いず,Bluetooth を用いたアドホック接続の通信性能を評価した. 図 3 に示すように Windows 端末と Android 端末間の相互の通信を測定した. これは Wi-Fi が使えない環境におけるアドホック通信を想定している.

3.2.3 モバイルクラウドを利用する通信

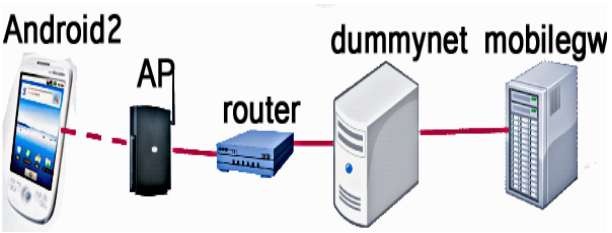

図 4 高遅延環境におけるサーバと Android の通信

図 4 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 Android 端末とサーバ（mobilegw）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携帯端末からアクセスする通信を想定している.

表 1　実験環境 (端末およびサーバ)

Ubuntu	OS CPU Main Memory	Ubuntu8.10(Linux2.6.27-7) iIntel(R) Pentium(R) M processor 1.73GHz 512MB
WindowsXP	OS CPU Main Memory Bluetooth	Microsoft Windows XP Intel(R) Pentium(R) 4 CPU 3.00GHz 512MB Bluetooth Ver2.1+EDR
mobilegw	OS CPU Main Memory	Fedora Core release 3 Intel(R) Pentium(R) 4 CPU 3.00GHz 1GB

表 2　実験環境 (Android 側)

Android1	Model number	HT-03A
	Firmware version	1.5
	Baseband version	6.2.50S.20.17H_2.22.19.26I
	Mod version	docomo standard
	Build number	CDB72
	CPU	Qualcomm MSM7201A 528MHz
Android2	Model number	T-Mobile myTouch 3G
	Firmware version	1.6
	Baseband version	2.6.29.6-cm42 shade@toyxygene
	Mod version	CyanogenMod-4.2.3.1
	Build number	DRC92
	CPU	Qualcomm MSM7201A 528MHz
	Bluetooth	Bluetooth Ver2.0+EDR

4. 性能測定

本研究では iperf を用いてソケット通信の性能を測定した. その結果を以下に示す.

4.1 Wi-Fi 環境における同一無線 LAN 内の通信

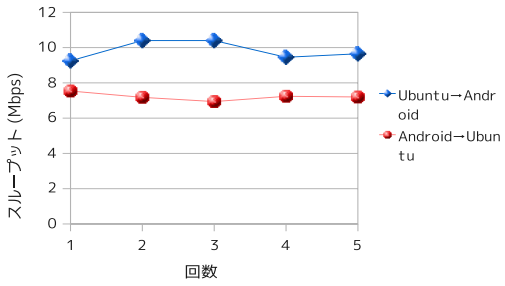

図 5　Wi-Fi を用いた AP 経由の TCP 通信

Ubuntu から Android への TCP 通信の平均スループットは 9.83(Mbps),Android から Ubuntu への TCP 通信の平均スループットは 8.56(Mbps) となった. 図 5 のグラフからわかる通り Android の方が送信性能が低いことがわかった. また UDP 通信は,Ubuntu から Android への平均スループットが 8.85(Mbps),Android から Ubuntu への平均スループットは 6.47(Mbps) となった. この結果は buffer size にスループットが依存しなかった.

また Android から Android への通信の場合,TCP 通信の平均スループットは 4.62(Mbps),UDP 通信は 5.88(Mbps) という結果になった.

4.2 Wi-Fi のない環境におけるアドホック通信

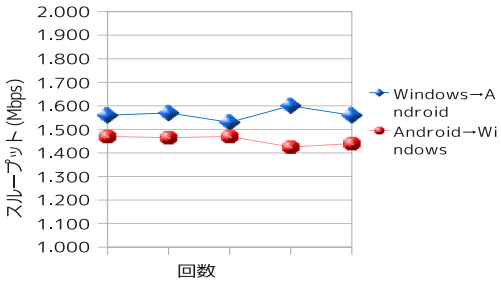

図 6　Bluetooth を用いた AP を経由しない TCP 通信

Bluetooth による通信も図 6 からわかるように Android の方が送信性能が低いことがわかった. しかし,Wi-Fi を用いた AP 経由による通信の結果よりも, スループットの差が小さいことがわかった. また TCP 通信においてスループットは Window size に依存しないことがわかった. Bluetooth を用いた UDP 通信のスループットの値は TCP 通信とほぼ同じ結果になった. さらに UDP 通信では Android から WindowsXP に送る場合 buffer size に結果は依存しないが Windows から Android に送る場合, 受け手である Android の buffer size が大きい方がスループットが高くなることがわかった.

4.3 モバイルクラウドを利用する通信

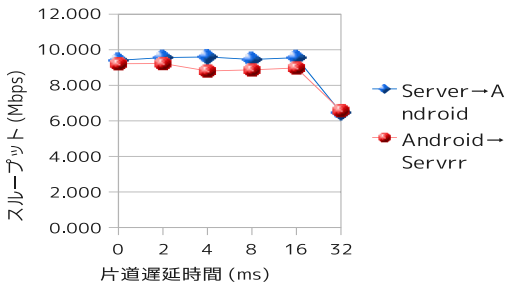

図 7　高遅延環境におけるサーバと Android の TCP 通信

遅延装置を使って, 片道遅延時間 0-32ms の高遅延環境を作り,Android と mobilegw 間のスループットを測定した. 結果は図 7 からわかる通り, 片道遅延時間が 16ms までは安定したスループットが得られるが, それ以上になると大幅にスループット

が低下してしまうことがわかった．また他の実験同様,Android の方が送信性能が若干低いことがわかった．

UDP における通信では mobilegw から Android への平均スループットが 7.92(Mbps),Android から mobilegw への平均スループットが 6.00(Mbps) という結果になった．UDP 通信では片道遅延時間が 32ms になってもスループットは安定していて低下しなかった．

5. TCP チューニング

次にモバイルクラウドを利用するケースにおいて TCP の輻輳制御アルゴリズムを変え，その性能の違いを測定した． 本研究のサーバ側で用いた輻輳制御アルゴリズムは,bic,cubic(default),htcp,reno,westwood の 5 種類である． TCP チューニングを行うためにサーバの OS を Fedora release 9(linux2.6.25-14.fc9.i686.PAE) にバージョンアップした． Android 側で用いた輻輳制御アルゴリズムは,reno,wertwood(default) である． Android のカーネルは linux2.6.29.6-cm-42 であるが，対応している輻輳制御アルゴリズムは 2 つのみであった．

5.1 輻輳制御アルゴリズム

bic は RTT 公平性に着目し積極的に window size を増加させるアルゴリズムである．cubic は bic の改良版で window size の増加が緩やかである．htcp はロスベースの高速 TCP であり，RTT や帯域幅が極端に大きい広域・高速ネットワーク上では，その帯域幅に合わせて効率的な振る舞いをする．reno は輻輳が発生したときの輻輳 window size の 2 分の 1 の値を ssthresh に設定し，後に輻輳が発生した場合は輻輳 window size を ssthresh から再開して輻輳回避段階に入るアルゴリズムである．これにより過剰な輻輳 window size の減少を避けられるため高速な転送を可能としている．westwood は reno に比べ，太い帯域で且つ漏れやパケットロスに強いアルゴリズムである．

5.2 サーバ側 TCP チューニング

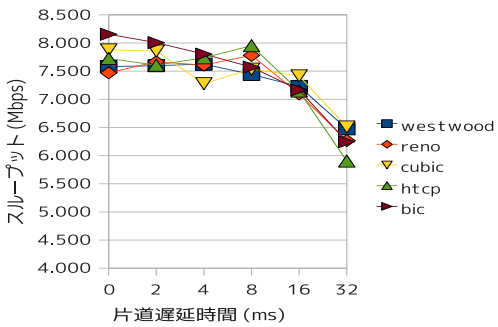

図 8 高遅延環境におけるサーバと Android の TCP 通信

図 8 にパケットロスのない環境でサーバに各アルゴリズムを適用した測定結果を示す．データ送信はサーバから Android 端末方向である．どの輻輳制御アルゴリズムも遅延時間が大きくなるにつれて性能の低下が確認された。片道遅延時間が 4ms までは bic アルゴリズムのスループットの性能が良く、cubic,westwood アルゴリズムは遅延時間が大きい時に、他のアルゴリズムに比べスループットは出ていることが確認された．

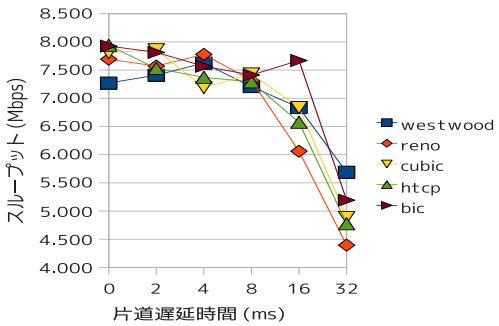

図 9 高遅延, パケットロス環境におけるサーバと Android の TCP 通信

図 9 は図 8 の環境に往復 0.2 %のパケットロスを与えた場合の結果である．パケットロスが加わった場合、片道遅延時間が 32ms の時の性能低下が著しい。また全体的に安定しない結果となった。パケットロスに強いとされる westwood アルゴリズムは他の 4 つのアルゴリズムに比べ一番スループットの振れ幅が狭く、遅延時間が 32ms の場合一番性能が高いことが確認された。

5.3 Android 側 TCP チューニング

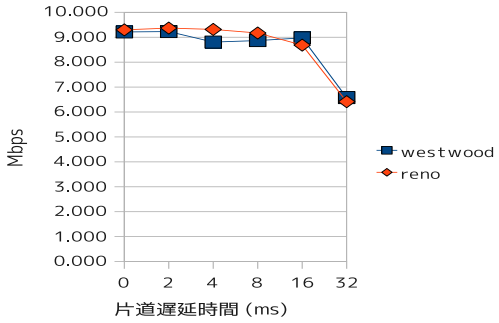

図 10 高遅延環境における Android とサーバの TCP 通信

図 10 に Android 端末に 2 種類の輻輳制御アルゴリズムを適用しサーバ方向へデータ送信を行った時のスループット測定結果を示す．

図 10 から遅延のみの環境では westwood,reno 共に性能に大きな差はなくほぼ同等の結果となった．またサーバから送出時よりも安定した高い性能が得られることが確認された。

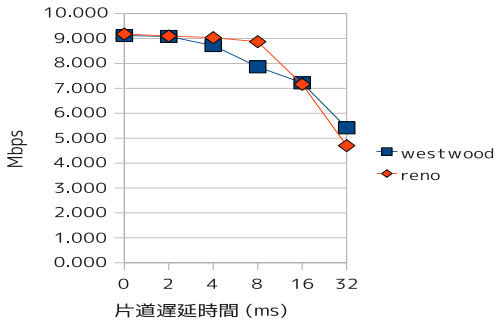

図 11 高遅延, パケットロス環境における Android とサーバの TCP 通信

図 11 は図 10 の環境に往復 0.2 %のパケットロスを与えた場合の結果である．パケットロスが加わった場合，片道遅延時間が 8ms までは reno の方が安定した結果が得られるが遅延が大

きくなるにつれて性能が大幅に減少していることがわかる. 一方,westwood は片道遅延時間が 4ms から性能が落ちてきているが reno ほど急激な性能の減衰は見られない.

図 10,11 から Android の default アルゴリズム reno よりも westwood の方が安定したスループットを得られることがわかった.

6. 実験環境における干渉問題

6.1 2.4GHz 帯における干渉

本研究では測定時に Android とアクセスポイント間を Wi-Fi 接続にて測定している. Wi-Fi で使われている周波数帯は Bluetooth と同じ 2.40GHz から 2.48GHz の周波数帯域「ISM RF（Industrial, Scientific, Medical Radio Frequency）帯域」であり、Bluetooth 動作時には常に WiFi のデータ転送速度が低下し、特に Bluetooth の送信機が近くにある場合には影響が大きいことが確認されている [5]. Wi-Fi は Bluetooth による影響を受けやすく, 半径 10m 以内に Bluetooth 搭載機器が稼働していた場合, スループットは半分程度まで減少してしまう.

5 章の結果は他の無線による干渉の影響が少ない深夜の時間帯に測定したものである. 同時刻に周辺で Android が検知できた Bluetooth 搭載機器はない.

Android アプリケーション,Wifi Analyzer によって取得したその時の Wi-Fi 状況を図 12 に示す.
本研究で用いたアクセスポイントは `001D7369D9C0_G` である.

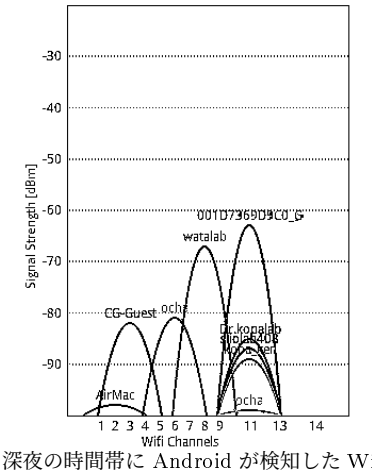

図 12 深夜の時間帯に Android が検知した Wi-Fi 状況

次に昼間の干渉が多い時間帯の Wi-Fi 状況を図 13 に示す. この場合, 近くに `001D7369D9C0_G` よりも強い `Wi-Fi` があるためスループットが低下することが確認された.

更にその他スループット低下の原因となる干渉が起こるケースとして, Android 端末における Wi-Fi,Bluetooth 関連アプリケーションを同時に利用した時, 2 台の Android を近くで実行し 1 台の同じサーバ間で通信を行った時, 周りで電子レンジを使っている時などが確認できた.

反対に Android をアクセスポイント付近に配置し測定したところスループットが上がることも確認した.

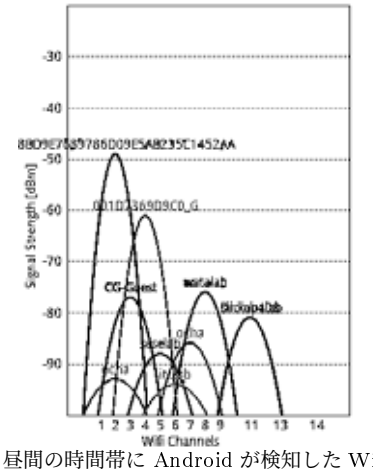

図 13 昼間の時間帯に Android が検知した Wi-Fi 状況

6.2 干渉環境における TCP チューニング

日常的な通信環境において, 周囲の干渉問題をさけて通信できることは少ない. より実用的な例として, 日中, 周りで Bluetooth 搭載機器が稼働しており, 他の Wi-Fi 干渉も受けている環境で同じ実験を行った. 結果を図 16〜17 に示す.

6.2.1 干渉環境におけるサーバ側 TCP チューニング

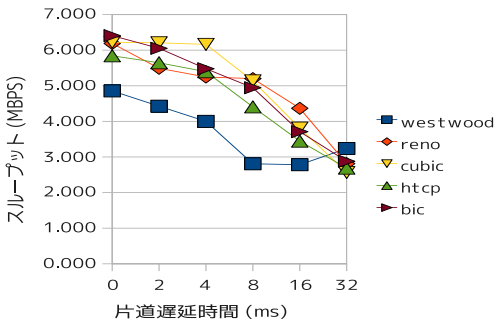

図 14 干渉あり高遅延環境におけるサーバと Android の TCP 通信

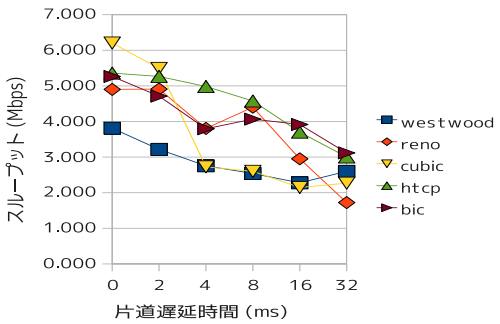

図 15 干渉あり高遅延, パケットロス環境におけるサーバと Android の TCP 通信

干渉がなかった場合, パケットロスなしの環境では各種輻輳制御アルゴリズムでの性能の大差がなかったが, 干渉を受けた場合は遅延のみでも性能にばらつきがある. これは, 輻輳制御アルゴリズムによるものではなく, 測定時の周りの Wi-Fi や Bluetooth の使用状況に左右されたものと考えられる. また全体的な性能も低下している. パケットロスが加わった場合, 更に性能が低下し測定値にばらつきが生じている.

6.2.2 干渉環境における Android 側 TCP チューニング

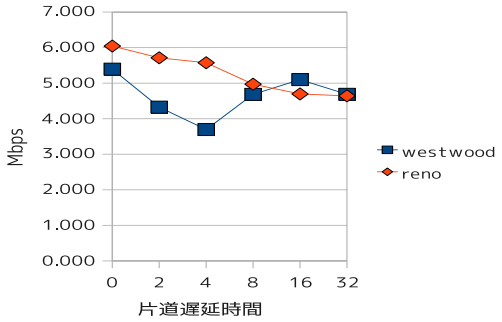

図 16　干渉あり高遅延環境における Android とサーバの TCP 通信

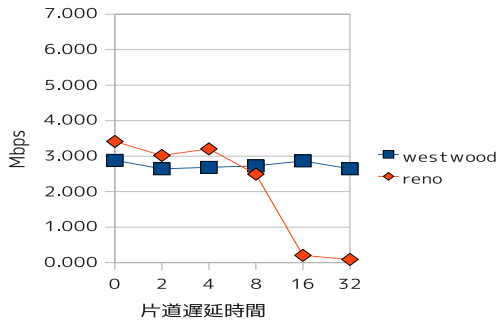

図 17　干渉あり高遅延, パケットロス環境における Android とサーバの TCP 通信

図 8〜11, 図 14〜17 から Android が送出する場合の方がサーバが送出する場合よりも干渉の影響を受けやすいといえる. 特にパケットロスを与えた環境では性能が半分程まで低下している.

その後の実験で近くに強い Wi-Fi があると性能が 2Mbps ほど低下することを確認した. また Android は Bluetooth による干渉を顕著に受け近くで Bluetooth 搭載機がペアリング可能な状態で稼働していた時（Android から検知できたとき）最も影響を受けやすい. また Android で Wifi analyzer など Wi-Fi 関連のアプリケーションを稼働させていたときや Bluetooth と Wi-Fi を同時使用しているときもスループットが低下することを確認した. さらに近くで電子レンジが稼働している場合もスループットの低下が確認された. 図 14〜17 に示す干渉時のグラフはあくまで一例であり, 再現性の無いもので, 測定時の周辺の環境に大きく影響される.

7. ま と め

Android における Iperf の測定ができる環境の構築を行い, 様々な通信ケースを想定し, それらのスループット測定を行った. 結果からそれぞれの通信ケースにおけるスループットが window size に依存するときとしないときがあることなど, それぞれの通信ケースの特徴がわかった.

また TCP の輻輳制御アルゴリズムを適用させた実験を行ったが, Android は輻輳制御アルゴリズムよりも特に送出時に Wi-Fi の干渉に影響されることがわかった.

8. 今後の課題

輻輳制御アルゴリズムによっては大きな性能の差が確認されなかったので今後は Android のカーネルの書換え等により, 干渉が発生する環境においても TCP 通信における性能向上を目指す.

文　献

[1] Android:http://www.google.co.jp/mobile/android
[2] Iperf:http://downloads.sourceforge.net/project/iperf/iperf/2.0.4
[3] Sourcery G++ Lite 2008q-3-72 for ARM GNU/Linux:http://www.codesourcery.com/, http://www.codesourcery.com/sgpp/lite/arm/portal/release644
[4] BUSYBOX:http://busybox.net/downloads/busybox-1.10.1.tar.gz
[5] Richard A Quinnell:周波数帯域でしのぎを削る WiFI と Bluetooth, http://ednjapan.rbi-j.com/content/issue/2005/11/content03.html