社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

Android端末を用いた無線通信時の制御ミドルウェアに関する考察

平井　弘実†　三木香央理†　山口　実靖††　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
†† 工学院大学　〒 163–8677 東京都新宿区西新宿 1–24–2
E-mail: †{hiromi,kaori}@ogl.is.ocha.ac.jp, ††sane@cc.kogakuin.ac.jp, †††oguchi@computer.org

あらまし　近年情報爆発の時代を迎え，スマートフォンが爆発的に普及している．スマートフォンは場所や時間を選ばずにインターネットを利用できる携帯端末であり，移動先や移動中において無線 LAN アクセスポイントに接続して通信を行うことが多いが，一台のアクセスポイントに同時に多くの端末が接続して通信を行おうとすると輻輳が発生する．本研究では，各端末同士が帯域を効率良く分け合うことで，無線 LAN 環境下において輻輳による通信性能の低下を緩和するミドルウェアの開発を目指している．本稿では環境に応じて TCP の輻輳制御手法を切り替えるミドルウェアの基盤となるソフトウェアを開発した．

キーワード　ネットワーク管理，ミドルウェア，TCP/IP，輻輳制御，Android

A Study on Transmission-Control Middleware on Android Terminal in a Wireless LAN Environment

Hiromi HIRAI†, Kaori MIKI†, Saneyasu YAMAGUCHI††, and Masato OGUCHI†

† Ochanomizu University　Otsuka 2–1–1, Bunkyo-ku, Tokyo, 112–8610 Japan
†† Kogakuin University　Nishishinjuku 1–24–2, Shinjuku-ku, Tokyo, 163–8677 Japan
E-mail: †{hiromi,kaori}@ogl.is.ocha.ac.jp, ††sane@cc.kogakuin.ac.jp, †††oguchi@computer.org

Abstract　Recently, Smartphone are becoming to be used explosively. In this study, we decided to develop middleware which allows for the efficient use of High-speed wireless LAN communication, in consideration of the number of terminals that share the access point.

In this paper, we show the basis of transmission-control middleware. The system shares each Android terminal's condition of communication and visualizes them.

In the future, we will establish new types of High-speed wireless LAN communication to develop middleware that selects TCP which is adaptable to the environment.

Key words　Network Management，Middleware，TCP/IP，Congestion control, Android

1. ま え が き

近年情報爆発の時代を迎え，場所や時を選ばずに気軽にコンピュータを使いたいという需要が高まり，モバイル端末であるスマートフォンが登場した．スマートフォンは携帯には優れているが，リソースやストレージが最小限であることから処理能力は劣るため，必要な時に必要なデータをサーバからインターネットを介してダウンロードして利用し，編集を終えるとアップロードしてサーバに保存する使い方が一般的である．またクラウドデバイスとしての役割も急成長しており，スマートフォンはこれからの時代にいつでもどこでも情報を発信するためのツールとして大いに期待されている．すなわち通信性能を向上させることによって，スマートフォンはもっと便利なものになると言える．そこで，本研究ではスマートフォンの性能向上のため，スマートフォンの通信システムに着目した．そして現在のシェア率も高く，オープンソースで開発が自由に行える Android OS を研究対象として採用した．

スマートフォンは携帯端末であるため，移動先及び移動中にさまざまなアクセスポイントに接続して通信を行う．しかし従来の携帯電話よりもマシンのスペックが高いため，公共の場の無線空間に大量のパケットが飛び交うようになると輻輳を引き起こしやすい．他の通信がないような条件の良い環境において，1 台のアクセスポイントを占有できた場合は高スループットが期待できるが，通常は 1 台のアクセスポイントや基地局に多く

のモバイル端末がつながっているため，他の通信デバイスと競合を起こしながら通信を行っている．一般に無線通信を行う端末は条件が悪くなるとパケットロスが頻繁に発生し，途端に性能が劣化してしまう．

このように競合している環境では，パケットの送り出しや受け取りのタイミングを制御するトランスポート層の役割が性能に決定的な影響を与えている． これまでの研究で，輻輳ウィンドウサイズと通信スループットの相関関係を可視化するツールを開発し，実際に実験を行ってみたところ，輻輳ウィンドウサイズが不安定になったとき必要以上に転送量を下げすぎていることが確認できた [7]．トランスポート層は輻輳ウィンドウで送り出す量を調節しているため，この値をできるだけ大きく保つことは望ましい．しかし，他の端末が送受信を行っている場合，単純に輻輳ウィンドウを最大値で固定した状態で複数端末が同時に無線通信を行うと，多数のパケットロスを発生させ，なかなか確認応答を受信できず却って転送速度は下がってしまう．

そこで，周囲の環境に応じた正確かつ適切なパラメータの切り替えが必要となる． 既存のモバイル向けトランスポート層プロトコルは，小さなバンド幅を効率的に分け合うことを目指して開発されたもので，低スループットかつ信頼性の低いものである．しかし，これからはモバイル端末においても大量データの高速無線通信を行う時代になりつつあるため，これに適した制御メカニズムを再構築する必要がある． 本研究ではモバイル環境においてカーネモニタに基づく適応的制御の手法を開発する．Android 端末で通信時のカーネル内のパラメータ情報を上位層から取得し，アクセスポイントを共有する端末間で相互に情報共有することで，通信状況に応じたフレキシブルな適応的制御を行い，高速無線通信のアクセス競合時に安定した高スループットを保つミドルウェアの実装を目標とする．

2. Android OS

2.1 アーキテクチャ

Android は，OS，ミドルウェア，アプリケーション，ユーザインターフェースをセットにしたモバイル端末向けプラットフォームであり，Google 社を中心として開発が行われている．図 1 に示すように，Android は Linux2.6 をベースとし，主にスマートフォンやタブレット端末をターゲットに，それらに適したコンポーネントが追加されている [2]．他の Linux OS と大きく異なる部分は，独自に開発された Android アプリケーション専用の Runtime である Dalvik 仮想マシンを搭載している点である．

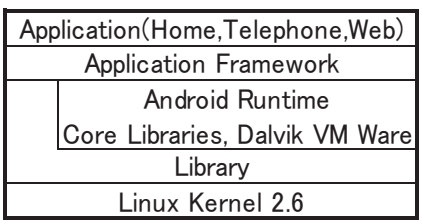

図 1　Android のアーキテクチャ

Android は Linux をベースとしているため，もちろんネイティブコードを実行することもできるが，Java 言語で書かれた Dalvik 実行形式のアプリケーションを利用することには多くの利点がある．Dalvik 仮想マシンを利用することにより，アプリケーションを移植性の高い環境で実行することができる．また Android アーキテクチャの上層部に存在するユーザインタフェースやフレームワークを利用できるため，直感的な操作を可能とする．

2.2 Android アプリケーション

Android は，開発環境においても無償で構築することができ，オープンソースである点からも対応アプリケーションが作りやすく数も増えるというメリットがある．また Android はキャリア間の制約がないため，アプリケーション開発においても自由度及び汎用性が高いだけでなく，一度マーケットに登録すると，世界中の Android ユーザからインストールが可能となる．現在 Android マーケットでは，このような大きなビジネスチャンスを提供されているため，毎年多くのアプリケーションが登録されており，アプリ市場は賑わっている．

Android マーケットの存在により，ユーザから見てもアプリケーションの入手は容易である．Dalvik 実行形式のバイトコードの状態で配布されているため，必要なアプリケーションをインストールして，スマートフォンを自由にカスタマイズできる．広告から収益を得ることによりアプリ自体は無償で提供されているものも多く，気軽にインストールして利用できる．

本研究はこれらのサービスを提供するシステムプラットフォームとしての Android に焦点を当て，通信システムの高速化を目指しているが，このように Android 端末においてアプリケーションの存在を無視することはできない．そこで本研究ではアプリケーションからの無線通信利用を前提として，通信スループットの高速化を目指す．

3. Android の TCP システム

3.1 輻輳制御アルゴリズム

Linux OS の通信において，カーネルのトランスポート層の輻輳制御が大きな役割を果たしている．TCP は送信側の送り出す転送量を調節する輻輳ウィンドウというパラメータを持っている．この輻輳ウィンドウとは，転送先からの確認応答を受信することなく，一度に連続して送り出せる最大のセグメント数を示す送信側のトランスポート層のパラメータである．高遅延環境においては特に通信スループットへの影響が大きく，輻輳ウィンドウが大きい時の通信スループットは高い値を示す．

輻輳制御とは，輻輳ウィンドウを適切な大きさに保つための制御である．TCP は正常な通信時には確認応答を受信するごとに，この輻輳ウィンドウを増加させるが，エラーが検出されるとネットワークに異常が存在すると判断し，輻輳ウィンドウを減少させる．[1] 輻輳ウィンドウが低下する原因は，送信側のバッファ溢れによるエラー，重複 ACK，SACK を受信した場合とタイムアウトを検出した場合が挙げられる．

Android のデフォルト輻輳制御アルゴリズムは TCP CUBIC である．CUBIC とは BIC の派生アルゴリズムであり，BIC ア

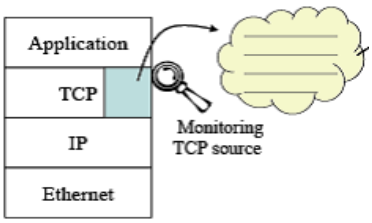

図 2　カーネルモニタ

ルゴリズムと同様の制御を三次関数状で行うアルゴリズムである．本稿ではこの TCP CUBIC アルゴリズムを利用して実験を行った．

3.2　輻輳ウィンドウサイズの取得

輻輳ウィンドウはカーネル内部のパラメータであるが，カーネルは通常のアプリケーションとは異なる特殊なソフトウェアであり，通常のアプリケーションのようなデバッグ手法は使えないため，汎用 PC においても，通信時の TCP の振舞を知る事は困難である．そこで本研究では，汎用 PC 用のカーネルモニタを Android に組み込み，輻輳ウィンドウの遷移を観察した. [3]

カーネルモニタとは，TCP の処理が行われるごとに各パラメータの値をログとして残すツールであり，いつ TCP のどの部分のコードが実行されているかを明らかにすることができる．図 2 に示すように，TCP のソースコードにモニタ関数を挿入し，カーネルを再構築することで，メモリからログを得ることが可能となる．カーネルモニタは輻輳ウィンドウの他にも，タイムスタンプ，ソケットバッファキュー長などのパラメータや各種エラーイベントの発生タイミングも取得することができる．

これまで出力したログを解析することにより実験を行ったが，Android 端末は CPU パワーやストレージ等のリソース量が制限されているため，汎用 PC と同じアプローチは困難であった．カーネルモニタはパラメータが切り替わるごとにログを残すため，解析には大量のメモリを消費する．実際に通信中に解析処理を並列で行ったところ，通信システムの動作を阻害してしまうことを確認した．そこで，今回は通信処理と解析処理を並列で実行するために，カーネルモニタを最新のログのみを出力するように対象となるソースコードの出力処理を書き換えて，Android に組み込んだ．

3.3　複数の Android 端末による同一アクセスポイント共有時の輻輳

ネットワークが混雑した際には，パケットの紛失や破損が頻繁に起こるので，エラーの発生確率を下げるために，輻輳制御によって輻輳ウィンドウサイズが急減し，通信スループットは下がる．しかしながら，パケットロスなどのエラーが必ずしも混雑を示すわけではない．一台のアクセスポイントを占有し通信を行っている時でもこのようなことは起こるため，帯域を余らせていることがある．このような状況において他端末の通信状況をパラメータとして加えた制御ができれば，これまでにない新しいネットワーク制御手法を確立することができると考えられる．本研究では競合時の通信スループット向上とともに，多端末共有時の通信公平性についても注目していく．

3.4　輻輳ウィンドウ制御ミドルウェア開発

本研究では，同一アクセスポイント共有下において輻輳ウィンドウサイズの最適化を行う制御ミドルウェアの開発を目標としている．

実際の利用形態として，Android 端末は広帯域有線ネットワーク接続されたクラウドのサーバと通信する場合が多く，無線の限られた帯域を通過できれば，その先の有線通信ではパケットが溢れてしまうことは考えにくい．つまり AP までスムーズに通信することができれば，その後は支障なく通信できる可能性が高いと考えられる．従って，その前提においては，アクセスポイントまわりの状況に応じて最適化を行うことが効果的であると考えられる．

そこで本研究は同一アクセスポイントを共有する無線 LAN 空間内において，互いの端末の通信状況すなわち輻輳ウィンドウを通知し合い，帯域を効率良く，平等に分け合う制御手法を開発する．既存の TCP 輻輳制御は ACK を受信するごとに輻輳ウィンドウを増加させるものであり，最大値まで上がりきって輻輳発生時に急減させる．特に高遅延環境においては輻輳ウィンドウは一度急減すると元の大きさに回復するまでに時間がかかり，それまでの間通信スループットが低下する．この間帯域を使いきらずに余らせてしまっているのは無駄であるため，これを避ける手法を開発する．

Android 端末は，様々な場所で利用することが想定できるため，状況に合わせた輻輳制御アルゴリズムを選び，適応させていく通信システム開発を目指す．

4.　通信制御方針

4.1　状況に合わせた制御アルゴリズム

TCP は汎用性が高くなるように実装されているため，その制御がどのような状況においても最適であるとは限らない．特に，同一アクセスポイント空間内で同時に 1〜2 台しか通信を行っていないという状況は頻繁に起こると想像されるが, デフォルトの TCP の制御は控え目に抑えすぎてしまうことがある．本稿では，同一アクセスポイント空間内に他端末が十分少ないケースを想定し，輻輳ウィンドウを大きく保つように加工した TCP を切り替え用に用意した．

図 3，図 4 では，往復遅延時間 128ms，人工パケットロス 0.5%を加えて，デフォルトの輻輳制御アルゴリズムと切り替え用に用意した輻輳制御アルゴリズムの振舞を比較する．これらは AP を占有した状態で 16Mbyte のパケットを転送した時の輻輳制御である．図 3 に示すように，デフォルトの TCP では輻輳ウィンドウを下げすぎているので，通信スループットが低いと考えられる．一方で独自の TCP はエラーにより輻輳ウィンドウが下がった時でも，すぐに元の値まで回復するように設計した．

このグラフは自作の可視化ツールによって作成したものである．縦軸が輻輳ウィンドウの大きさを示し，横軸が時間で単位は秒である．この独自の TCP により，往復遅延 64ms 以上の高遅延環境で有用であることが既に確認されている [5].

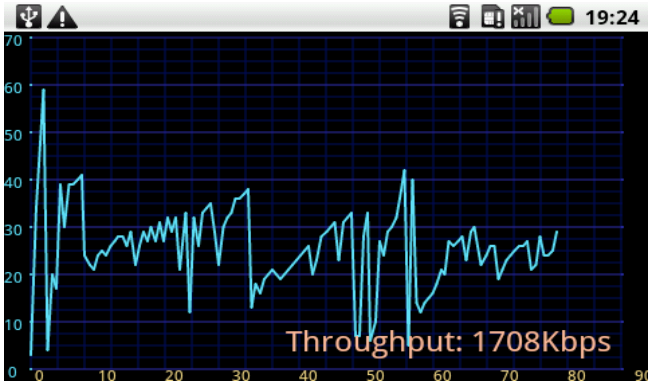

図 3　Default の輻輳制御

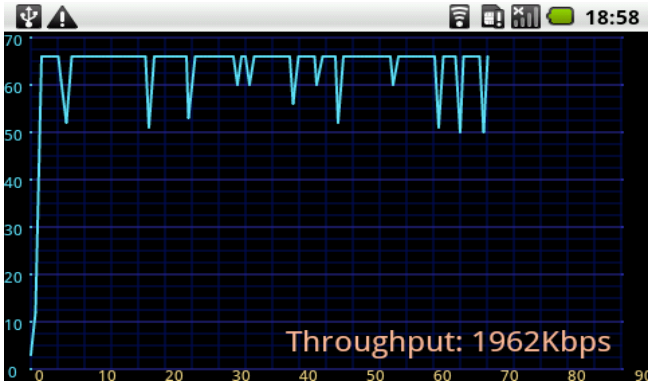

図 4　独自 TCP の輻輳制御

また，本研究の以下の実験では Google 社による文献 [6] を参考に Initial Congestion Window Size(輻輳ウィンドウサイズ初期値) を 10 に変更し，通信スループットの向上と安定を図った．また輻輳ウィンドウサイズ初期値を 10 にすると，遅延がない状態で立ち上がりの通信スループットが 1.25 倍程向上することを確認した [4]．TCP がスロースタートを行う際は極めて短時間で輻輳ウィンドウサイズを増加させ，輻輳ウィンドウサイズはすぐに 10 に達するため，輻輳ウィンドウサイズ初期値を 10 にすることの弊害は生じない．また輻輳ウィンドウは送信側におけるパケット送出レートの自主制限であるため，送信側の制御を独自のものに変更したとしても，受信側との間で互換性の問題が生じる心配はない．

5.　他端末の輻輳ウィンドウ共有システム

本稿では環境に応じて TCP の輻輳制御手法を切り替えるミドルウェアの基盤となるソフトウェアを紹介する．現在はこのソフトウェアの大部分は Dalvik 仮想マシン上で動いているが，将来的にミドルウェアとしてライブラリに埋め込む予定である．

5.1　端末上のアプリケーション

このソフトウェアにより指定した大きさのパケットを転送し，その通信スループットを測定できるようになったが，さらに改良し，転送中の輻輳制御の振舞を可視化できるようにした．実験を行うシステムは，図 5 のように設置する．

5.2　ソケット通信

本実験装置では，PC 上で 2 種類のサーバを利用する．一方は

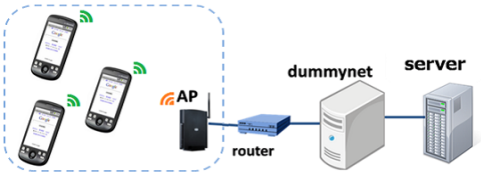

図 5　通信システム概要

指定された大きさのパケットを受信するサーバであり，もう一方は通信状況の情報共有のためのサーバである．転送パケットを受信するサーバは指定サイズのパケットを受け取りきると応答を返す．この応答により，Android 端末側は転送を始めた時刻と応答を受け取った時刻を比較することで，通信スループットを算出することができる．ここで計算した通信スループットは，パケット転送完了時に画面の右下に文字列で表示する．

そしてこの通信コネクションが確立されている間は，並列処理でカーネルモニタの情報取得が随時行われている．本実験で利用するカーネルモニタは Android 端末用に出力を最新のログのみに抑えたものであるため，定期的に読み出すことが必要となる．ここでは 800ms の sleep() 処理を挟み，無限ループでログの解析を行った．

5.3　通信状況の情報共有

自端末の TCP 解析については 5.2 章に述べたが，ここではその情報を知らせあう機能について述べる．各端末は自分の TCP の解析をしてから，タイムスタンプと輻輳ウィンドウのみを取り出して，情報共有用のサーバに転送する．情報共有用のサーバはパケットを受信するサーバと同一 PC 上で実行しているが，異なるポートを指定することにより，分離させている．情報共有用のサーバは受け取った情報をコネクションを確立している端末にブロードキャストする．端末ごとの識別するために今回は IP アドレスを利用したが，本システムは同一アクセスポイント空間内での利用を想定しているため，MAC アドレスを利用することもできる．

ブロードキャストされた情報は，各 Android 端末に転送され，各 Android 端末上では自分の TCP を行いながら他端末の情報も同時に可視化する．本稿では手動で TCP の切り替えを行ったが，今後はプログラムが自動で状況を判断して切り替えを行うようにする．また現在は PC 上のサーバを利用しているため，サーバを介して情報共有を行っているが，これは分散システムの概念から常時サーバを必要とするのは，汎用性が低くなってしまうため，今後はピア・ツー・ピアのソケット通信で，各自に通信状況を知らせ合う分散型管理を実装することを予定している．

5.4　実験環境

本実験で使用した実験環境を表 1 に示す．尚，人工遅延装置には FreeBSD の dummynet を利用した．

5.5　実験内容

独自の輻輳ウィンドウ共有システムを用いて実験を行った．このシステムで利用するアプリケーションは同一アクセスポイントを利用している Android 端末間で互いの通信状況を通知

表 1　実験環境

Android	Hardware	HT-03a
	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
Server	CPU	Intel Pentium M 1.30GHz
	Main Memory	256MB
	OS	Linux2.6.32-31-generic
AP	Maker	BUFFALO
	Product name	WHR-G301N/U AirStation
	Mode	IEEE 802.11g

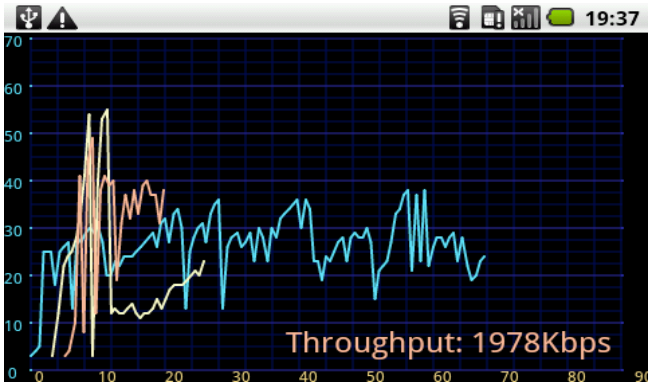

図 6　実験 1 の実行結果

し，各端末で可視化している．

本実験では 1 台のアクセスポイントに対し，3 台の Android 端末を用意した．1 台の Android 端末が 16Mbyte のパケット転送を行っている間に，その他 2 台の Android 端末が途中から加わり 4Mbyte のパケット転送を行う．往復遅延時間は 128ms，人工パケットロス 0.5%とした．

TCP の切替えを確認するために 2 つの実験を行った．実験 1 では全ての Android 端末の TCP がデフォルトの状態で，通信実験を行った．実験 2 では 16Mbyte のパケット転送を行う Android 端末のみを，他 2 台の転送処理が完了し，アクセスポイントを占有できた時に，TCP を独自のものに切り替え，その後も通信を続けた．

5.6　実験結果と評価

これらの実験結果を図 6，図 7 に示す．グラフの縦軸は輻輳ウィンドウ，横軸は時間である．実験 1 では，2 台の Android 端末が 4Mbyte 転送を完了させてからも 2 度の輻輳が発生している．また輻輳が発生し，輻輳ウィンドウを急減してから，元の値に回復するまでに帯域を余らせている．

一方実験 2 では，2 台の Android 端末が転送を完了させてから独自の TCP に切り替えたため，輻輳ウィンドウは落ちることなく，安定した通信を再開することができている．

6.　まとめと今後の課題

本稿では，同一無線空間において通信状況を示すパラメータ

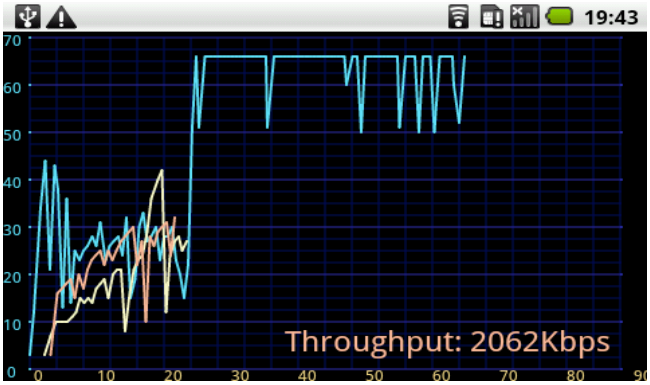

図 7　実験 2 の実行結果

を通知し合うシステムを開発し，他端末の通信状況の可視化に成功した．また状況に応じて TCP を切り替えることにより通信スループットを上げられることを確認した．

本稿の実験で既存の TCP とは別に用意した輻輳制御アルゴリズムは，高遅延環境の通信において既存の TCP よりも効率が良いことが確認されている．今後は同一無線空間で通信を行っている他端末の情報に応じて最適なアルゴリズムを選択するために，自作の輻輳制御アルゴリズムの種類を増やす．また他端末の情報を利用して，自動で TCP を切り替えられる仕組みを開発する．

謝辞　本研究を進めるにあたり，ご指導して下さった株式会社 KDDI 研究所の竹森敬祐さん，磯原隆将さん，株式会社 NEC Technologies の Iain Williams さんに深く感謝致します．

文　献

[1] W.Richard Stevens 著，橘康雄，井上尚司訳，詳解 TCP/IP Vol.1 プロトコル，ピアソン・エデュケーション，東京，2000.
[2] android developers:http://developer.android.com
[3] 三木香央理，山口実靖，小口正人:Android 端末におけるカーネルモニタの導入，Comsys2010，2010 年 11 月.
[4] 三木香央理，山口実靖，小口正人:カーネルモニタを用いた Android 端末の無線 LAN 通信時の通信性能の考察，DEIM Forum2011，C6-1，2011 年 3 月.
[5] 三木香央理，山口実靖，小口正人:カーネルモニタを用いた Android 端末の無線 LAN 通信性能の解析と性能向上のための一検討，DICOMO2011，7H-2，2011 年 7 月.
[6] Nandita Dukkipati, Tiziana Rece, Yuchung Cheng, Jerry Chu, Tom Herbert, Amit Agarwal, Arvind Jain, and Natalia Sutin, ”An Argument for Increasing TCP ’s Initial Congestion Window ”ACM SIGCOMM Computer Communications Review, vol. 40 ,No.3, pp. 27- 33,July2010. http://research.google.com/pubs/pub36640.html
[7] 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，情報処理学会第 73 回全国大会，5V-9，2011 年 3 月.
[8] 平井弘実，三木香央理，山口実靖，小口正人:無線環境下における Android 端末の通信制御ミドルウェア構築に向けた一検討，DICOMO2011，7H-3，2011 年 7 月