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

平井弘実[†1]　三木香央理[†1]
山口実靖[†2]　小口正人[†1]

近年，携帯電話に代わる端末としてスマートフォンが爆発的に普及している．スマートフォンは移動端末であり，移動先や移動中において様々なアクセスポイントに接続して通信を行うため，一台のアクセスポイントを共有するスマートフォンデバイスが多い時，輻輳により通信性能の低下が起きている．

本研究では，スマートフォンをより便利な端末にするために通信性能無線 LAN 環境下において，実際に同一アクセスポイントの共有端末数を考慮した発信側の TCP パラメータ調節を行い，効率のよい高速無線通信の確立するミドルウェアの開発を目指している．そこでオープンソースで自由に開発が行える Android 携帯を研究対象として取り上げた．

本稿では既存のシステムの課題を明らかにするため，さまざまな Block Size で通信実験を行い，それぞれの通信スループットを測定した．ここでは BlockSize とは，送信バッファをアプリケーション層からトランスポート層に渡す時に分割する大きさを指す．

またこの実験結果を元に，代表的な BlockSize でパケット転送を行っているときの TCP の振舞を解析を行った．この解析を元に，BlockSize 別の通信性能，及びスループット低下の原因となるフロー制御を観察し考察する．

今後は同一アクセスポイント共有ノード数を考慮して，トランスポート層の送り出すセグメント量を最適化するミドルウェアを開発し，より高性能な通信の確立を目指す．

A Study for Developing TCP Controlling Middleware on Android Terminal in a Wireless LAN Environment

Hiromi HIRAI,[†1] Kaori MIKI,[†1]
Saneyasu YAMAGUCHI[†2] and Masato OGUCHI[†1]

In recent years, Smartphones has become extremely popular on behalf of the cell phone. due to insufficient processing power, we often use Smartphone with Cloud service or Web services. However, Smartphone is a mobile terminal, so we are used to changing from an access point to another access point while we move. When many Smartphone terminal share an access point on the same time, Smartphone hardly accesses to the Internet because of congestion.

In this study, for Smartphone being more convenient, we decided to develop a middleware which helps High-speed wireless LAN communication efficiently, considering the number of terminals sharing the access point. We selected Android OS as research material because it is OpenSource so we can develop freely.

In this paper, in order to clarify the problems of the existing system, we measured throughput of different BlockSize transmissions. We call [the buffer size which Application layer passes to Transport layer] BlockSize.

Based on the results of this experiment, we also analyzed typical BlockSize of packet transmission. From these analysis, we consider the relation between transmission throughput and congestion behavior. In the future, we will establish new type High-speed wireless LAN communication to develop a middleware which control congestion window that is amount of segments TCP send considering the number of nodes sharing the same access point.

1. はじめに

近年情報爆発の時代を迎え，気軽にいつでもどこでも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の影響を受けやすい．他の通

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

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，確かに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，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. 研 究 目 的

2.1 輻 輳 制 御

Android のアーキテクチャは，Linux カーネルをベースとしている．この Linux カーネル内部のトランスポート層の輻輳制御が, 通信において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．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．

2.2 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り，ユーザ空間から取得することはできない．そこで本研究では，オリジナルツールであるカーネルモニタを利用して，輻輳ウィンドウの遷移を観察した．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 1 に示すように，TCP のソースコードに挿入し，カーネルコンパイルを行うことで，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．本研究では，通信状態を解析するために，このカーネルモニタを組込み機器である Android に応用した.[4]

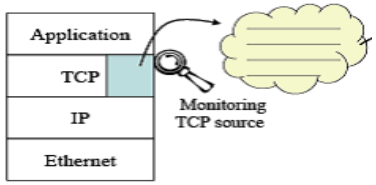

図 1 カーネルモニタの仕組み

2.3 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，輻輳制御によって輻輳ウィンドウサイズが急減し，通信性能は低くなる．しかしながらこのような状況

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．

2.4 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．

現在の通信回線は品質の良いものとなり，有線ケーブル内やルータにおけるパケットロスはほとんどを起こらなくなったため，エンドーエンドの通信のうち主に無線 LAN 空間におけるパケットロスがボトルネックであると言える．つまり無線 LAN 空間内における工夫ができれば大変有効である．

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知らせ合い，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のスループットが低下する．しかし，ここで他ノードの通信状況がわかっていれば，最大値まで上げきらずに適切な輻輳ウィンドウの大きさを保つことにより，無駄な輻輳発生を未然に防げる可能性があり，安定したスループットを得られると考えられる．

また本研究では制御手法は，既存の TCP に手を加えず，アプリケーション層とトランスポート層の間で機能し，適切なブロックサイズに分割したり，タイミングを調節することで TCP が最適な振舞をするように制御し，安定した高スループットを持続する通信システムを構築する．

3. 複数台の Android 端末を用いた性能測定実験

アプリケーション層から渡される BlockSize の大きさによって TCP がどのような振舞をするか明らかにするため，複数の Android 端末から 1 台のアクセスポイントを共有して無線 LAN 空間からパケット転送を行う実験を行った．

この実験では，Android 端末側がサーバに対してソケット通信要求を行う．コネクションが確立されたら，合計 16Mbyte のパケットを指定した BlockSize に分割し，トランスポート層に渡して転送を行う．ソケット通信を行うプログラムは自作したものを用いる．[7] この通信プログラムは Android アプリケーションで実装しているため，ここで得られるスループットは Android 端末の Dalvik VM 上からのパケット転送による通信スループットである．Android アプリケーション上の通信スループットを測定対象としたのは，ハードウェア独立な端末に依存しない汎用の通信手法の確立を目指すためである．また Android 端末の機能はほとんどがアプリケーションによって操作されているため，Android アプリケーション上で行われるインターネット通信を想定できる点でも，Dalvik VM 上の通信スループットを得ることが有効だと考えられる．

Android 端末は無線 LAN を利用してアクセスポイントを経由するが，この時に複数台の端末が同一無線 LAN 帯域を共有しているため，共有する台数ごとに性能は大きく異なる．本稿ではこれらの環境における通信スループットの比較を行った．

測定用アプリケーションからは合計 16Mbyte のパケットを転送するが，パケットを分割する BlockSize の最大値は 8Mbyte とした．なぜなら 8Mbyte より大きな BlockSize を使って転送を行うとアプリケーションが強制終了してしまうため，これ以上の BlockSize を転送することはないと考えられるからである．

Android 端末上のアプリケーションは指定した一定量のパケットを転送し，サーバプログラムはパケットを一定量受信しきると応答を返す．そして Android アプリケーションはサーバからの応答を受け取るまでの時間を測定し，転送したパケット量を時間で割ることによりスループットを求めた．

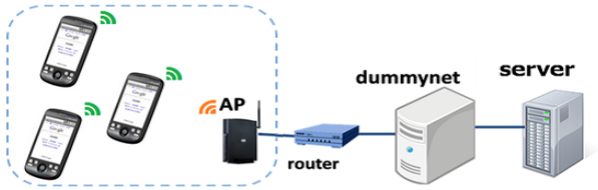

図 2 実験概要

3.1 実験環境

本実験を行った環境を表 1 に示す．実験に利用した Android 端末の輻輳ウィンドウの初期値は 10 と設定した．Google 社から配布されるオリジナルの Android OS よりも通信スループットを向上させられるという既存研究に則り，このように変更を加えた．[5][6]

Android 端末は一台の無線 LAN アクセスポイントに IEEE802.11g で接続し，アクセスポイントはルータと人工遅延装置である Dummynet を介して PC サーバに繋がっている．

表 1　Experimental Environment

Android Device	Hardware	HT-03a
	Model number	AOSP on Sapphire(US)
	Firmware version	2.1-update1
	Baseband version	62.50S.20.17Hpsy”5F2.22.19.26I
	Kernel version	2.6.29-00481-ga8089eb-dirty
	Build number	aosppsy”5Fsapphirepsy”5Fus-eng 2.1-update1 ERE27
Server Terminal	CPU	Intel Pentium M 1.30GHz
	Main Memory	256MB
	OS	Linux2.6.32-31-generic
Access Point	Maker	BUFFALO
	Product name	WHR-G301N/U AirStation
	Mode	IEEE 802.11g

Dummynet は FreeBSD を利用して PC 上に構築し，64ms の往復遅延を加えた．高遅延環境では輻輳ウィンドウとスループットの相関性が高くなるが，図 3 に示すように，128ms 以上の高遅延環境では，3 台以下の端末数ではスループットに差が見られない．そこで今回は最も高遅延であり，台数ごとの違いが見える 64ms の遅延を加えて実験を行った．

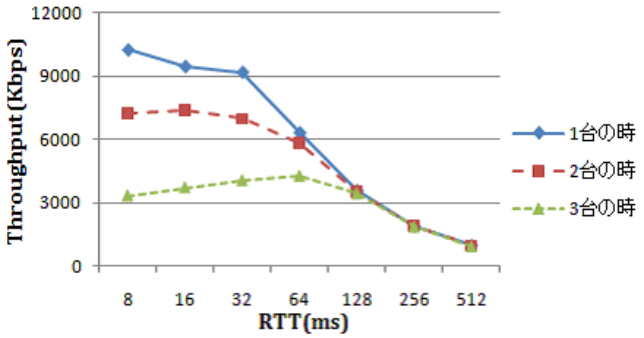

図 3　往復遅延時間別スループット
転送パケット:16Mbyte，BlockSize:128Kbyte

3.2 実験結果

複数の Android 端末による同一アクセスポイントを共有したパケット転送を行った実験結果を図 4 に示す．同一アクセスポイントに Android 端末 1 台だけが接続する時は，アクセスポイントを占有できるため，高いスループットを保持しているが，6Mbps 程度で一旦飽和する．さらに BlockSize が 128Kbyte〜512Kbyte の時は最高のスループットに到達するが、2Mbyte 以上になるとスループットが低下する．

Android 端末が 2 台の時は競合しているが，BlockSize が 2Kbyte 以上の時は，5.8Mbps 程度の安定したスループットを維持している．しかし BlockSize 512Kbyte と 8Mbyte の時は，2 台の端末間に 500Kbps 程の差が生じていて公平な通信ではない．

最後に Android 端末が 3 台の時は，1 台及び 2 台の時に比べてスループットがとても低く，また不安定であることがわかる．1 台の時と同様に，BlockSize が 8Mbyte の時はスループットが低下する． 次に TCP の解析を行った結果を以下の図 5〜13 に表す．同じ環境で

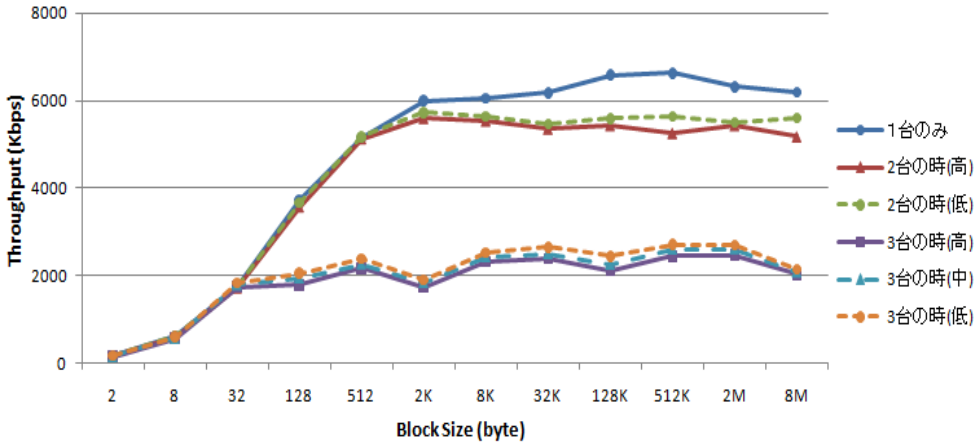

図 4　BlockSize 別スループット

も TCP の振舞は毎回少しずつ異なるが，ここでは典型的な例として再現性のある遷移を取り上げた．

3.2.1 同一アクセスポイントに Android 端末 1 台のみが接続している時

1 台のアクセスポイントに Android 端末 1 台のみが接続している時の輻輳ウィンドウの遷移を図 5〜7 に示す．BlockSize が 128byte の時と 8Kbyte の時は輻輳は発生せず，とても安定した通信を行っている．しかし 8Mbyte になるとスループットが低下する．8Mbyte

で転送した時は，Android の輻輳ウィンドウの最大値である 66 に達することもあり，輻輳ウィンドウが大きくなり過ぎたことによるバッファ溢れがスループット低下の原因だと考えられる．

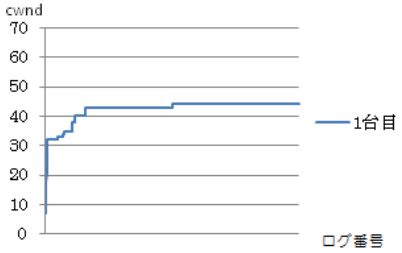

図 5 同一アクセスポイントに 1 台のみの接続で BlockSize:128byte

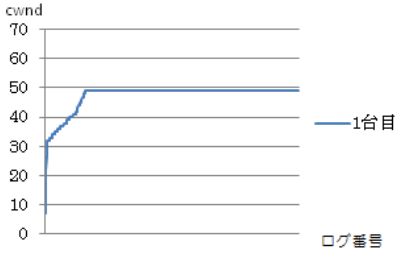

図 6 同一アクセスポイントに 1 台のみの接続で BlockSize:8Kbyte

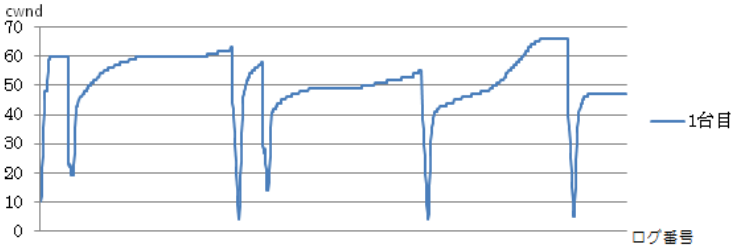

図 7 同一アクセスポイントに 1 台のみの接続で BlockSize:8Mbyte

3.2.2 同一アクセスポイントに Android 端末 2 台が接続している時

1 台のアクセスポイントに Android 端末 2 台が接続している時の遷移を図 8〜10 に示す．ここでは同一無線 LAN 帯域を分け合って通信を行うことになるが，常に公平に帯域を分けあっているとは言えない．図 8 に示すように BlockSize=128byte では 1 台目の Android 端末だけに輻輳が発生し，またその後輻輳ウィンドウが元の大きさに回復するまでに時間がかかっている．図 9 の BlockSize=8Kbyte では，1 台目の方が若干輻輳発生が多いが，輻輳ウィンドウがほぼ同じ大きさを保たれていて輻輳発生時もすぐに回復する，図 10 では 1 台目だけが輻輳ウィンドウサイズが最大に何度も達しているのに対し，2 台目は最大値まで上がりきることはなく，また輻輳発生確率も高い．

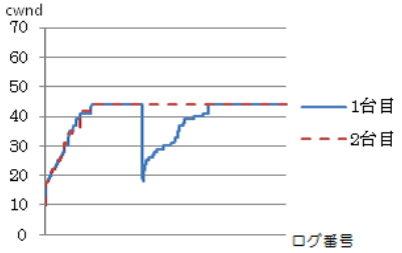

図 8 同一アクセスポイントに 2 台を接続して BlockSize:128byte

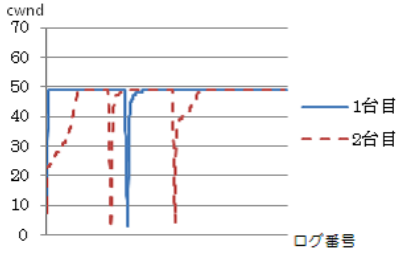

図 9 同一アクセスポイントに 2 台を接続して BlockSize:8Kbyte

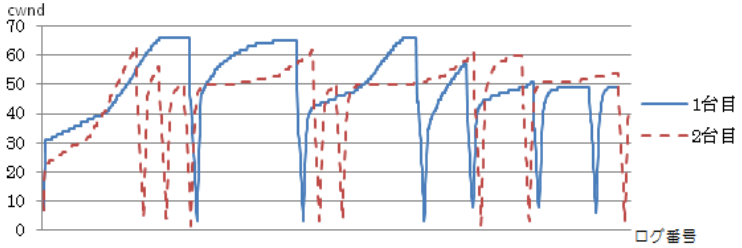

図 10 同一アクセスポイントに 2 台を接続して BlockSize:8Mbyte

3.2.3 同一アクセスポイントに Android 端末 3 台が接続している時

1 台のアクセスポイントに Android 端末を 3 台接続している時は，どの BlockSize においても輻輳発生率が高い．BlockSize=8Kbyte では図 11 に示すように輻輳ウィンドウが最大値まで上がりきらないのに対し，8Mbyte では図 13 に示すように，2 台目，3 台目は輻輳ウィンドウが高い値まで上がり切るのが早いが，輻輳発生時の輻輳ウィンドウの回復が遅く，また 1 台目の端末だけが輻輳ウィンドウを大きくすることができず，不公平な状態になっている．対して 8Kbyte の時は図 12 に示すように 3 台ともほぼ均一な輻輳ウィンドウ

の大きさを保持していて，輻輳ウィンドウが落ちてもすぐに元の大きさに回復していてとても安定している．

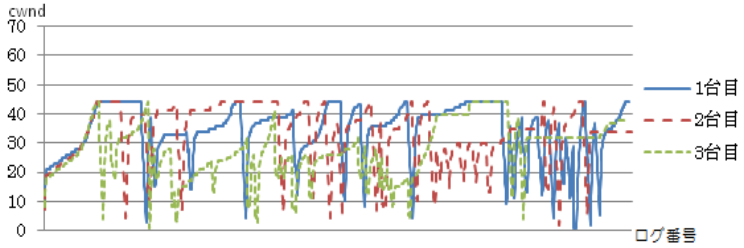

図 11 同一アクセスポイントに 3 台を接続して BlockSize:128byte

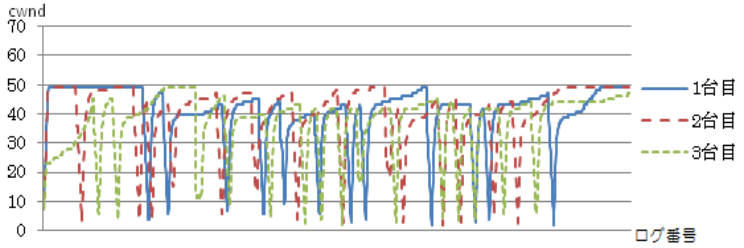

図 12 同一アクセスポイントに 3 台を接続して BlockSize:8Kbyte

3.3 考　　察

アクセスポイントを 1 台が占有している時は，BlockSize は 128Kbyte〜512Kbyte が最適値と言える．BlockSize を 2Mbyte より大きくすると図 7 のように，アクセスポイント占有時にも関わらず輻輳ウィンドウ低下が数回発生しているため，大きな BlockSize を受け取った際には，512Kbyte 以下に分割すれば，安定したフロー制御に保つことができ，スループットも向上させることができることがわかる．

Android 端末数が 2 台の時は，32byte〜2Kbyte の間で緩やかにスループットを上げていき，2Kbyte〜128Kbyte の間が高スループットで飽和している．端末数 2 台においても 512Kbyte〜8Mbyte の間では不公平性を解決する必要がある．

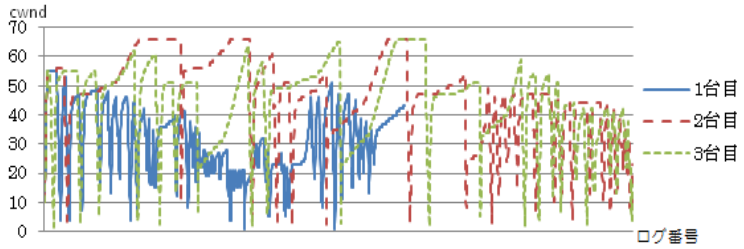

図 13 同一アクセスポイントに 3 台を接続して BlockSize:8Mbyte

3 台の時は 8Mbyte ではそれ以下の時よりもスループットが下がってしまうため，ここでも BlockSize の分割により TCP の振舞を安定させることで，スループットを向上させることができることがわかる．

解析結果を見ると輻輳発生確立が低いという点では BlockSize=128byte の時が最適のように思われる．しかし，128byte では実際に高スループットにはならないので，実用的ではない．

BlockSize=8Kbyte 時は端末 3 台の時においても高スループットかつ，全ての端末が公平に輻輳ウィンドウを分けあっていて，輻輳発生時もすぐに輻輳ウィンドウが元の大きさまで回復するため，理想的な通信と言える．本稿よりも多い端末数や他のノードが同一アクセスポイントを共有する際にもこの時のように輻輳制御になるように制御するミドルウェアを開発することが望ましい．

4. まとめと今後の課題

本稿では，輻輳ウィンドウ制御ミドルウェア開発を念頭に，TCP の既存のアルゴリズムをうまく利用する方法としてアプリケーション層がトランスポート層に送信バッファを渡す際の BlockSize の違いによるスループットの差，及びその通信中のフロー制御を観察し，考察を行った．BlockSize が大きすぎると輻輳発生確率が高くなり，無断の多い通信となってしまうこと，また複数端末競合下で BlockSize を環境に応じた最適な値にすることで TCP に手を加えず，アプリケーションからの制御により，安定したフロー制御を保てることがわかった．

今後は，環境ごとに異なった理想的な輻輳制御の振舞をより精密に明らかにするための実験を重ねる．そしてノード数を考慮に加えて理想的に TCP が振舞うように BlockSize を変

動させていくミドルウェアを開発し，より効率の良い公平な通信の確立を目指す.

参 考 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，2000
2) 小口正人， コンピュータネットワーク入門-TCP/IP プロトコル群とセキュリティ，サイエンス社，2007
3) android developers:http://developer.android.com
4) 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，Comsys 2010，2010 年 11 月
5) Nandita Dukkipati, Tiziana Refice, 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
6) 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
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