Android 端末を用いた周辺端末からの情報に基づく協調的制御手法の提案

平井弘実[†1]　山口実靖[†2]　小口正人[†1]

近年スマートフォンが爆発的に普及している．またスマートフォンの高機能化が進み，スマートフォンを用いた大容量高速通信に対する需要が大変高くなっている．本研究は，このような需要に対応するため，スマートフォンに最適な新しいネットワーク通信制御の仕組みを提案する．現行の輻輳制御アルゴリズムは，有線通信志向で開発されており，各端末が独立して，送出するセグメント数を見積もっていたため，ノイズの多い無線通信においては，最適な制御を行っているとは言い難く，課題が多く残されていた．そこで本研究は，同一アクセスポイントを共有している周辺端末を連携させ，互いの情報を利用して，より精密な可用帯域の見積もりを試みる．本稿では，一部の端末が過度に帯域を確保することのないよう，環境に応じて自主的にトラフィックの発生量を抑えることで公平性を向上させる仕組みを示し，ミドルウェアが，どのように制御するのが適切であるかを考察する．

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

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

In recent years, smartphone is spread worldwide. Along with the tendency, the performance is also being improved and the demand for mass high-speed communication is increasing. In this study, so as to meet the demand, we propose a transmission control mechanism in smartphone network. The present congestion control algorithm of TCP has been developed for wired network and does not always work in the best due to noise in wireless route. It is because each mobile terminal estimates available bandwidth independently only by transmission error rate. Therefore, we propose a new system in which each mobile terminal exchanges its transmission condition and estimates more precisely how many segments it should send for an ACK. In this paper, we clarify the way the middleware should control TCP with the maximum size of congestion window.

1. はじめに

近年，日本独自仕様の携帯電話に代わり，スマートフォンが爆発的に普及し始めている．従来の携帯電話は電話・電子メールに加えて低トラフィックなインターネットアクセスが可能であったが，スマートフォンは小型コンピュータという位置付けであり，多くの機能が実現された．従来の携帯電話では，OS に組み込まれた唯一のメーラやブラウザしか利用できなかったが，スマートフォンでは，メーラやブラウザに関しても自分の気に入ったアプリケーションをインストールして，利用形態に合うようにカスタマイズすることができる．また，スマートフォンは常にユーザによって携帯されるという点から，リアルタイムな情報のやり取りにおいても需要が高い．Twitter や Facebook などのソーシャルメディアにおいて，個人による近況報告が現在一般的になったが，今後は音声配信や動画配信などといった大容量データのやり取りが個人のスマートフォンによってなされる可能性が高いと考えられる．すなわち，これまでの携帯電話のインターネットアクセスにおいては，サーバ側からのデータのダウンロードが主であったが，今後はアップロード方向の通信も多くなって行く可能性が高い．近い将来には，上り通信フローにおける輻輳が重要な課題となると考え，本研究では，周辺で同じアクセスポイントを共有して通信を行っているスマートフォン端末間の連携による協調的制御手法を提案する．

スマートフォンはコンピュータと比較すると，そのリソースは不十分であるため，クラウドサービスを利用して，頻繁に無線アクセスを行う．クライアント・サーバ間通信は，クライアントのモバイル端末から身近なアクセスポイントまでの無線通信と，アクセスポイントからサーバまでの有線通信で繋がっているが，このようなネットワークトポロジでは，通常ボトルネックとなるのは，低帯域かつノイズの多い無線通信区間である．またスマートフォンは移動体であるため，ユーザの移動によって，一台のアクセスポイントに繋がっている端末数も変わりやすく，トラフィックに変動がある．このようにスマートフォンのネットワークは，これまでの有線接続を主とする PC 端末の通信とは性質が大きく異なり，また近年スマートフォンの大容量高速通信の需要が急増しているため，新しいネットワーク制御の仕組

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

みが必要であると言える．

本研究は，クライアント・サーバ間通信において，クライアント側からのパケット送信の際に，クライアントのアクセスポイント周りで，通信状態に関わるカーネル内部のパラメータを含む情報を互いに知らせ合うことにより，輻輳を回避し，最適な通信環境を実現する制御を行おうとする試みである．この概念図を図 1 に示す．既存研究においても，モバイル端末間で連携して通信の最適化を行うという手法はこれまでになく，本研究が初の試みと言える．これまでのモバイル端末はリソースが最小限であったため，大きな負荷に耐えられず，端末間で高度な制御を行う手法は現実的ではなかったが，近年スマートフォンの需要が高まり，ハードウェアのスペックが急速に向上したため，このような他端末と連携した制御が実現可能になりつつある．このような制御手法は，あらゆる無線端末で有効であると考えられるが，本研究では，シェア率が最も高く，オープンソースで自由に開発が行える Android 端末をターゲットとする．

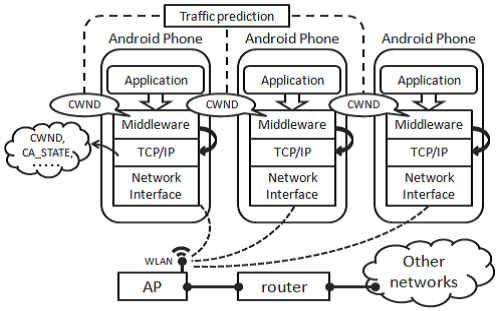

図 1　提案手法の概念図

2. Android OS

スマートフォンには，様々な種類が存在するが，その中でも一際注目を集めているのが Android OS である．Android OS とは，Google 社が提供する OS，ミドルウェア，アプリケーション，ユーザインタ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る．Android OS は，オープンソースで提供されているため，メーカやキャリアの制約がなく，様々なデバイスに自由に応用することができる．また端末メーカの視点では，標準的な OS が存在することにより，システムの開発費用を大幅に削減することができることが大きな利点である．これらの点から，Android を採用する端末メーカが増え，現在 Android はトップシェアとなった．

2.1　アーキテクチャ

図 2 に示すように，Android は Linux2.6 をベースとし，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[1)]．他の Linux OS と大きく異なる部分は，独自に開発された Android 用の Runtime である Dalvik 仮想マシンを搭載している点である．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となるため，アプリケーションは Dalvik 仮想マシンに合わせて開発すれば，直感的な操作性に優れた UI を利用することができ，移植性も高い．

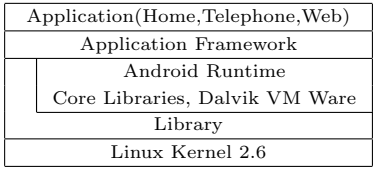

図 2　Android のアーキテクチャ

2.2　Android アプリケーション

Android アプリケーションの開発環境は，全て無償で提供されるソフトウェアで構築することができ，操作性に優れた多くのユーザインタフェースが用意されているため，知識の浅い開発者でも気軽に本格的なアプリケーションを開発することができる．さらに Google Play というアプリケーションマーケットに開発したアプリケーションを登録すると，世界中の Android ユーザが購入可能となるため，大きな収益を上げられる可能性がある．近年様々なアイデアが Android アプリケーションとして開花している．その中には，個人が現在の状況を知らせるといったソーシャルメディアも多く，今後は，個人が動画配信を行ったり，PC との連携で大規模データの同期を行ったりすることにより大容量通信の需要が大きく向上すると考えられる．

3. 輻輳制御の既存研究

3.1　輻輳制御の概要

輻輳とは，通信トラフィックが帯域やネットワーク機器の許容量を上回ることによって，パケット損失率が高くなる状態である．パケット損失が起こると，同じパケットを再送する必要があるので，通信スループットは低下してしまう．輻輳制御とは，データ送信側が帯域の混雑具合を予測し，輻輳を未然に防ぐための制御である．TCP では，輻輳ウィンドウと

いう，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パラメータの大きさを調節することで，輻輳が起きないように制御している．

特に高遅延環境においては，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くなれば，連続して送り出せるセグメント数も多くなり，通信スループットは高くなる．つまり，より賢い制御を行い輻輳が起きない範囲で輻輳ウィンドウを大きく保つことができれば，通信スループットは向上する．

これまでも輻輳制御アルゴリズムに関する多くの研究がなされているが，それらの大部分がシミュレータを使った理論的な測定であり，実機の端末で実測値を得た研究は少ない．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 に組み込むことで，実機の輻輳ウィンドウの遷移を観察した[6]．Android 端末の実機を用いた通信制御に関する研究は，まだほとんど行われていない．

3.2 カーネルモニタと通信制御可視化ツール

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，TCP のどの部分のコードがいつ実行されているかを明らかにすることができる．図 3 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．

カーネルモニタを用いて出力したログを解析することにより汎用 PC においては実験を行うことが可能であるが，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る．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仕様であったため，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，通信処理と解析処理を並列で実行するために，カーネルモニタが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．

さらに本研究では，カーネルモニタによって得られたログを解析し，輻輳ウィンドウの値をブロードキャストするミドルウェアにおいて，発信されたデータを受信し，時系列にそってグラフに描画するアプリケーションを開発した．これは著者らの開発した Android の通信システム可視化アプリケーションを拡張したものである[9]．

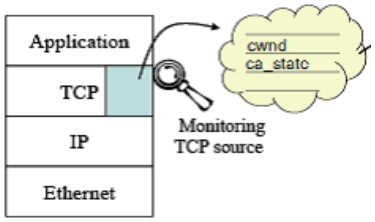

図 3 カーネルモニタ

3.3 現行の輻輳制御アルゴリズム

これまでに研究開発が行われ実装された TCP の輻輳制御アルゴリズムは，遅延ベース方式，損失ベース方式，及びそれらのハイブリット方式に分けることができる．遅延ベース方式とは，データ転送中に計測された RTT の実測値と理論値を比較し，輻輳ウィンドウの大きさを調節する制御手法である．遅延に基づく正確な制御であるが，損失ベース方式との競合によるスループットの低下を招くというデメリットがある．損失ベース方式は，正常な確認応答を受信したら輻輳ウィンドウを増加させ，エラーを検出すると減少させる制御である．有線通信においては，通信エラーは一般に経路の混雑によるルータのバッファ溢れを示すため，この手法はシンプルかつ効果的であるが，無線通信においては，ノイズによる通信不良も帯域の混雑とみなしてしまうため，未使用帯域を大きく余らせてしまうことがある．

Android のデフォルト輻輳制御アルゴリズムは，損失ベース方式を採る TCP-CUBIC である．CUBIC とは BIC (Binary Increase Control) の派生アルゴリズムであり，BIC のアルゴリズムと同様の制御を三次関数状で行う．

図 4〜6 は，表 1 に示す Nexus S 端末にカーネルモニタを組み込み，通信中の輻輳ウィンドウの振舞を観察したものである．これは，Nexus S 端末とサーバ端末の間に人工遅延装置を設置し，人工遅延及び人工パケット損失を加えた時の通信中における輻輳ウィンドウの遷移を示す．人口遅延装置とは，FreeBSD[3] の Dummynet を利用し，任意のネットワーク環境を構築する PC 端末を示す．解析結果 1，2 は，1 台の Nexus S 端末が単独で，アクセスポイントを占有し，Dummynet により通信経路上に人工パケット損失率をそれぞれ 0%，0.2%加えた時の通信中の輻輳ウィンドウの遷移を解析したものである．一方で解析結果 3 では，パケット損失率 0%の環境下で 1 台にアクセスポイントに対し，4 台の Nexus S 端末

表 1　実験環境

Nexus S 端末	Model number	GT-I9023
	Firmware version	2.3.4
	Baseband version	I9023XXKD1
	Kernel version	2.6.35.7
	Build number	GRJ22
サーバ端末	OS	Ubuntu10.04.3 LTS
	Linux version	2.6.32-37-generic-pae
	CPU	Intel(R) Core(TM) i3 CPU 530 @ 2.93GHz
	Main Memory	3GB
Dummynet	OS	FreeBSD 6.4-RELEASE
	CPU	Intel(R) Pentium(R) 4 CPU 3.20GHz

を同時に接続し，通信を行った時の輻輳ウィンドウの遷移を解析したものである．

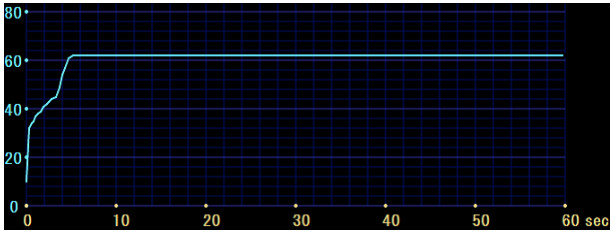

図 4　輻輳ウィンドウの解析結果 1:人工遅延 32ms，人工パケット損失率 0%の環境下における単独通信

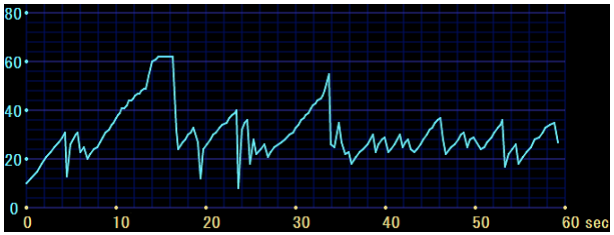

図 5　輻輳ウィンドウの解析結果 2:人工遅延 32ms，人工パケット損失率 0.2%の環境下における単独通信

解析結果 1 と解析結果 2 を比較すると，人工パケット損失率 0%の環境下では，輻輳ウィンドウが一定値まで増加するのに 5 秒程度かかるが，その後は安定した振舞をしている．解析結果 2 の通信経路には，Dummynet により，人工パケット損失を加えているが，解析結果 3 では，人工パケット損失の代わりに，他 3 台の端末が競合して同時に通信をすること

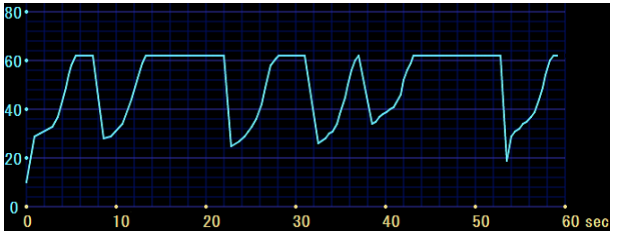

図 6　輻輳ウィンドウの解析結果 3:人工遅延 32ms，人工パケット損失率 0%の環境下における競合通信

で，ネットワークに負荷を掛けている．解析結果 1 と比較すると解析結果 2，3 は，輻輳ウィンドウの昇降状態が続き，安定していない．しかし，ここで解析結果 2 と解析結果 3 のパラメータ遷移が大きく異なるのは,[8] より，実験機における輻輳制御では，パケットのビットエラーや順番の入れ替えなどのパケット損失に比べて軽度な障害は，エラーと認識せず，ウィンドウ更新をしないためと考えられる．

4. 通信制御ミドルウェア

4.1 Android 端末間の情報連携

これまで無線環境における輻輳制御アルゴリズムの改善に関する研究開発が多くなされてきたが，PC 端末においては，基本は Ethernet の有線通信志向で実装されており，無線通信手段はあくまで，オプションにすぎなかった．しかし，スマートフォンは，有線通信手段を持たないため，完全に無線通信に特化した仕組みを導入することができる．現在スマートフォンにおいて，より大容量のデータをより高速に転送できる仕組みが求められているため，スマートフォンのネットワークを高性能化する新しい仕組みは大変興味深い．また，従来の携帯電話では，メモリなどのリソースが少なく複雑な仕組みは高負荷となってしまったが，近年のスマートフォンデバイスは高性能であり，今後さらに処理能力が向上すると考えられる．本研究はこのような背景に基づき，無線通信端末に適した通信制御ミドルウェアの提案を行う．

スマートフォンは，周辺にある基地局などのアクセスポイントを検出し，無線接続を試みる．Skype や Line などの P2P 通信もあり得るが，大部分が図 7 に示すようなサーバアクセスである．従来の輻輳制御アルゴリズムは，正確な経路情報を把握せずエンド・エンドで送出するセグメント数を調節してきた．しかし，有線区間と無線区間では大きく性質が異なるため，この手法は非効率である．有線区間のパケット損失は，トラフィックがネットワー

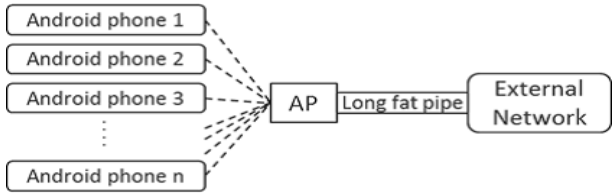

図 7 Android 端末のネットワークトポロジ

ク機器の許容量を上回っていることを示したが，無線区間においてはノイズの影響による可能性が十分にある．この問題を解決するため，Congestion Notification といった，実際の輻輳をオプションによって通知することで，帯域の見積もりをより正確にする手法が存在するが，これは経路上のネットワーク機器の対応が必須であるため，導入コストが大きく，実際には普及が滞っている．また近年のスマートフォンは，ノイズによる輻輳ウィンドウサイズの減少を抑えるために，わずかなパケット損失をエラーと判断しないアグレッシブな転送方法が実装されているケースも存在し，これは機種間の不公平な通信を招くだけでなく，輻輳崩壊を引き起こす可能性がある．

スマートフォンは転送量が限られているため，広帯域な有線区間において，輻輳崩壊が発生する可能性はほとんどなく，ボトルネックとなるのは，主に無線区間だと言える．そこで，本研究は，1 台のアクセスポイントに繋がった同じ無線区間で通信を行う端末同士が互いの情報を交換することで，より精密な帯域見積もりを行う手法を提案する．

4.2 Android 端末の基本性能測定

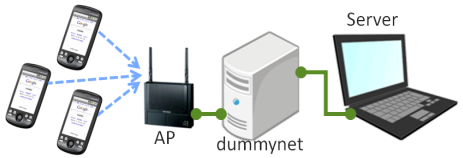

図 8 測定環境

現行の輻輳制御アルゴリズムは，各端末が独立してエンド・エンドの通信状況に基づき可用帯域を見積もっているため，不公平や非効率が生じている可能性がある．そこで，アクセスポイント 1 台に対し，複数台の Android 端末が接続され，競合して通信を行う際に，どのような影響が出るのかを調べた．

図 8 に示すように，アクセスポイントを複数台の端末で共有し，無線アクセスを行う．またアクセスポイントからサーバの間の有線経路上に，Dummynet を設置し，各遅延環境における通信スループットを測定した．また本研究では，各端末の不公平について注目するため，各実験で通信スループットの昇順にプロットしたグラフを図 9〜13 に示す．

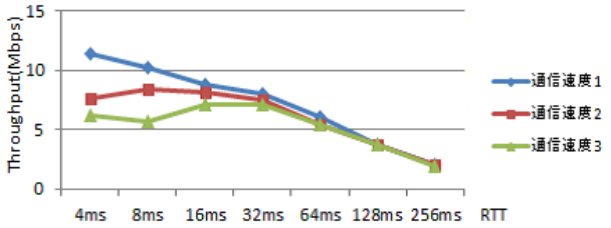

図 9 通信測定 1: 3 台同時通信時の通信スループット

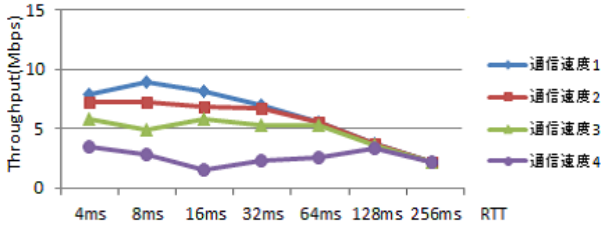

図 10 通信測定 2: 4 台同時通信時の通信スループット

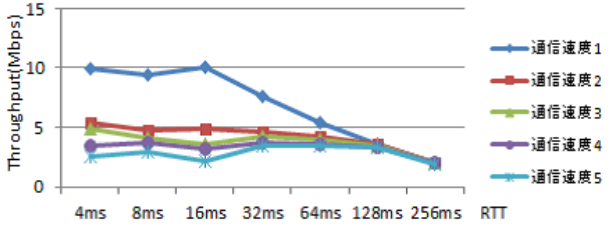

図 11 通信測定 3: 5 台同時通信時の通信スループット

これらの測定結果からわかるように，Android 端末は複数台で無線アクセスを行った場合に，各端末の通信スループットにばらつきがある．比較的端末数の少ない 3〜4 台の競合環境でも，最も高い値を示した端末の通信スループットと，最も低い値を示した端末の通信

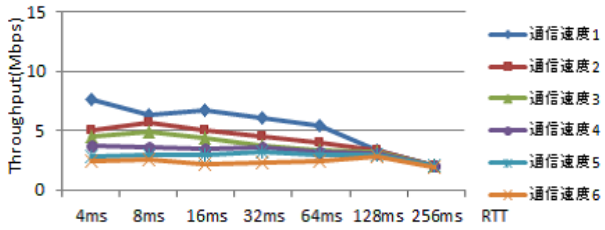

図 12 通信測定 4: 6 台同時通信時の通信スループット

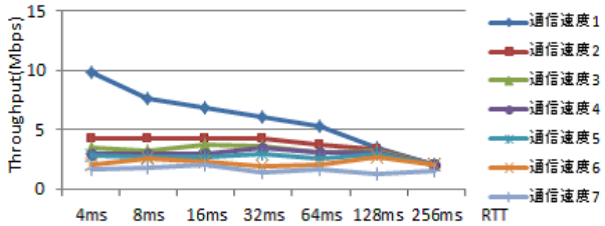

図 13 通信測定 5: 7 台同時通信時の通信スループット

スループットは 2 倍近く異なる．また端末数 5 台以上の環境では，それ以上の大きなばらつきがある．とりわけ図 11，13 においては，最も通信スループットの高い端末が帯域を過度に確保していることがわかる．これは，一度特定の端末の輻輳ウィンドウが高く上昇した時に，トラフィックが混み合い，他端末が帯域を確保しにくくなってしまったことが原因と考えられる．標準の TCP では，このような公平でない通信が成立してしまうため，このような点を克服できるような新しい仕組みは価値が高い．

5. 本研究が提案するミドルウェアの通信制御手法

本研究では，輻輳制御アルゴリズムを環境に応じて，外部プロセスからチューニングすることを試みる．TCP-CUBIC の輻輳制御アルゴリズムは，シンプルな設計となっているため，必ずしも全ての環境で最善であるとは言い切れない．しかし，他のアルゴリズムを用いるには，TCP-CUBIC との親和性を保つことが必須となるため，高速かつ親和性の高いアルゴリズムが必要となる．しかし，本研究では，Android カーネルの標準 TCP を利用しつつ，環境に応じてアルゴリズムの積極性を調節するため，親和性を保ちながら，高速な通信を実現できると考えられる．

まずミドルウェアによる環境に応じた通信制御を行うため，カーネルが通信処理を行っている間に，外部プロセスから輻輳ウィンドウの上限値を設定できる proc インタフェースを実装した．このインタフェースにより，通信中においても，同じアクセスポイントを共有する他端末が通信を始めたことをミドルウェアが検知すると，輻輳ウィンドウの上限値を環境に合った値に設定することで，トラフィック発生量を制限し，途中から通信を始めた端末も均等に帯域を分け合えるように制御する．図 14 は，輻輳ウィンドウの上限値を固定することで，どのように端末の通信能力を抑えることができるかを明らかにするために，各値で 1 台のみの端末の通信スループットを測定した結果である．このグラフから，輻輳ウィンドウの上限値が 50〜70 においては，大きな違いは見られないが，40 以下に設定した時は，通信スループットを抑えることができたことを確認できる．我々は，既に異なる機種が混在する環境における通信の不公平の克服のために，ミドルウェアで高機能端末が輻輳ウィンドウの上限値を抑え伝送レートを下げることで，低機能端末が帯域を確保できることを確認した[8)7)]．しかし，近年 Android のバージョンアップが進み，Android2.1〜Android4.0 が主流となった．そこで本稿では，Android2.3 が搭載された Nexus S 端末による公平性の向上を試みる．将来は，Android4.0 以降やメーカの独自仕様の違いを吸収し，公平に通信が行えるように応用したいと考えている．

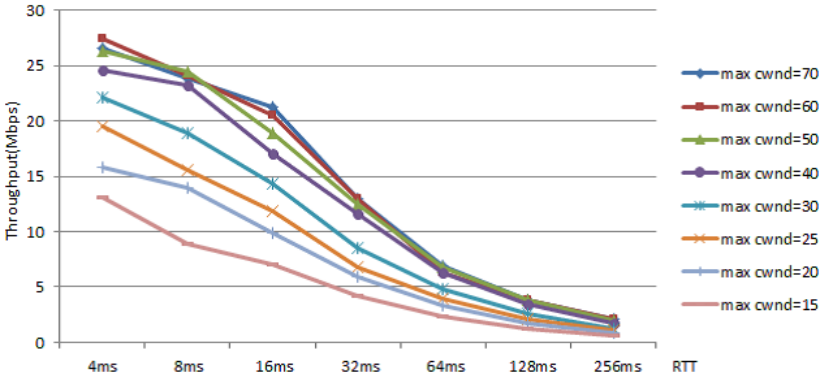

図 14 輻輳ウィンドウの上限値を固定した時の通信スループット

5.1 通信制御ミドルウェアの狙い

本研究では複数端末の競合通信時に公平性を確保する通信システムの確立を目指している．本稿では，ミドルウェアがどのように制御すれば，通信の高速化および公平化が図れるかを明らかにする．このような基礎実験において，高速性・公平性の向上を確認し，それら

に必要な条件を明確化した上で，今後は，環境に応じてフレキシブルにアルゴリズムを最適化するミドルウェアを開発する．

複数台の Android 端末が同時に通信を行っている間，通信スループットにばらつきがあることを 4.2 章で示した．そこでこれらのばらつきの原因は，一部の端末が輻輳ウィンドウの値を上昇させすぎていることであると仮定し，輻輳ウィンドウの上限値を環境ごとの適切な値に設定し，帯域を過度に確保してしまうことがないようにすることで公平な通信の実現を検討する．また，この方法により，輻輳ウィンドウが一定値以上に増加しないため，パケット損失の可能性を下げることができる．即ち，パケット損失により，輻輳ウィンドウを大きく減少し，スロースタートフェーズからパケット損失前の状態に戻るまでに，送り出せるセグメント数が限られてしまうという問題も解消できる可能性がある．図 6 においても，一度パケット損失が発生すると，ウィンドウサイズの回復までに 5 秒程度の時間がかかっているため，パケット損失が発生しないように輻輳ウィンドウを最適な値で留めることができたら，通信速度の向上が期待できると考える．

5.2 実　　験

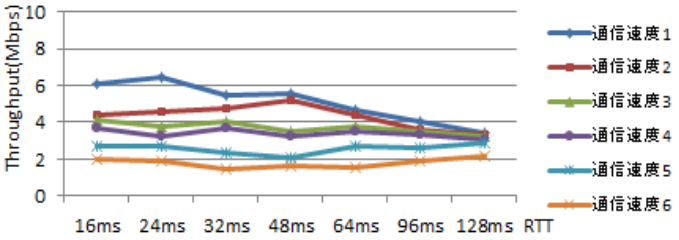

図 15　実験 1:標準 TCP を使用した 6 台の Android 端末が，同時に通信を行った時の通信スループット測定結果

輻輳ウィンドウの上限値を設定することにより，図 14 のように，各端末は割り当て帯域幅以下にトラフィック発生量を抑えることができる．そこで，本稿では，6 台の Android 端末が同時に通信を行う際に，輻輳ウィンドウの上限値を設定した時の効果を明らかにする．帯域をより多くのセグメントで埋めつつも，パケット損失が起こらない状態を維持するために，各遅延時間ごとの帯域幅遅延積を算出し，端末数で均等に割った分だけを各端末が使用できるようにし，各実験を行った．

実験 1 では，標準の TCP が動作する Android 端末 6 台を用いて，独自の通信制御をすることなく同時に通信を行い，通信スループットを測定した．そして，実験 2 では，パケット損失が起こらない通信を目指し，各端末のトラフィック発生量を帯域幅遅延積を均等に割り振った分しか帯域確保ができないように輻輳ウィンドウの最大値を設定し，同様の実験を行った．また，実験 3 では，パケット損失をある程度防止できる範囲の制御として，輻輳ウィンドウの最大値を実験 2 で設定した値の 2 倍に設定した Android 端末を用いて同様の実験を行った．実験 1〜3 の実験結果を，図 15〜17 に示す．

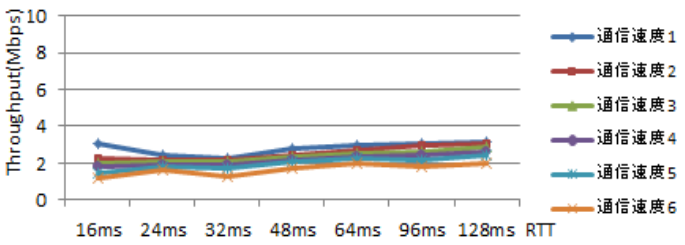

図 16　実験 2:パケット損失が発生しないよう，トラフィック発生量を制限した 6 台の Android 端末が，同時に通信を行った時の通信スループット測定結果

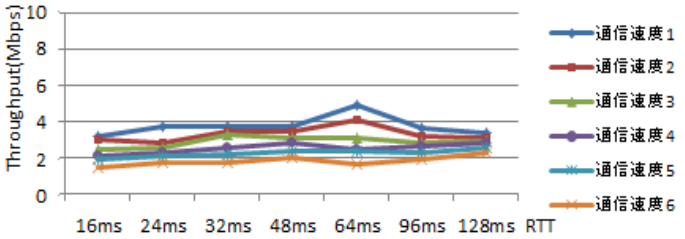

図 17　実験 3:輻輳ウィンドウの最大値を実験 2 で設定した 2 倍として，トラフィック発生量を制限した 6 台の Android 端末が，同時に通信を行った時の通信スループット測定結果

5.3 考　　察

5.2 節の実験結果の公平性を測る指標として，Jain の Fairness Index を用いる[5]．FI の値 (fi) は，以下のように定義される．本実験結果の Fairness Index を図 18 に示す．Fairness Index は，1.0 に近い程，公平な値であることを示す．

$$fi = \frac{(\sum_{i=i}^{k} x_i)^2}{k \sum_{i=i}^{k} {x_i}^2} \qquad (1 \leq i \leq k)$$

実験 1 では，標準 TCP を利用している Andorid 端末がそれぞれ独立してエンド・エンドの通信状況のみに基づき，可用帯域を見積もり，帯域確保をしているため，通信スループットにばらつきがある．一方で実験 2 では，トラフィック発生量を制限した Android 端末を利用しているため，全体としてはばらつきが大きく抑えられ，Fairness Index は，1.0 にかなり近い値となっており，公平性が大きく向上していることが明らかである．しかし，実験 2 の各々の Android 端末の通信スループットに焦点を当てると，通信性能が低下していることがわかる．公平性確保のためには，ある程度トータルスループットが減少することはやむを得ないが，実験 2 は高速化という点では課題が残る．今回の制御手法では，帯域幅遅延積を均等に配分した量のトラフィックしか送出できないため，特定の端末が帯域を取りすぎることは防げたものの，伝送レートを抑えすぎていたことが原因だと考えられる．また実験 3 では，パケット損失が起こりにくいように輻輳ウィンドウの最大値を設定したため，公平性をある程度確保した上で，高速性も大きく劣っていない．すなわち，ある程度の輻輳ウィンドウの増減を許しつつ，できるだけ転送エラーを防ぐように制御することが望ましいと考えられる．

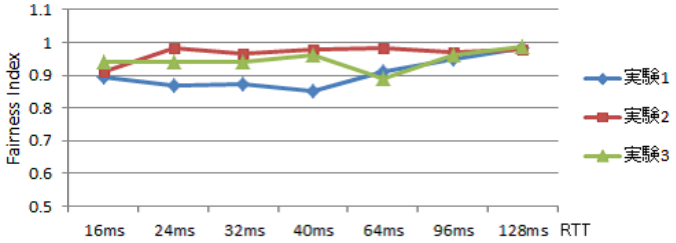

図 18 フェアネスインデックス

6. ま と め

本稿では，複数台の Android 端末が，1 台のアクセスポイントを共有し，競合した状態で通信を行った場合，各端末の通信スループットにばらつきがあることを明らかにした．通信制御ミドルウェアにより，環境に応じてフレキシブルに各端末のトラフィック発生量を制限することを念頭に，いくつかの基礎実験を行った．トラフィック発生量を抑えるために，輻輳ウィンドウの上限値を固定できる proc インタフェースを開発して，これによって端末の伝送レートを操作できることを明らかにし，実際に複数台の Android 端末が同時に通信を行っている時に，このような制御を行うと公平性が向上することを示した．

また，輻輳ウィンドウの上限値を固定しただけでは，帯域確保を行う端末の通信能力を過度に制限し，帯域を残すようにできたものの，最も確保できる帯域が少ない端末の通信スループットをを向上することはできなかった．これには様々な理由が考えられるが，輻輳ウィンドウの上限値だけではなく，下限値や ssthresh を操作して，輻輳ウィンドウが下がった状態が続く端末を救済することが有効だと考える．このようなより効果的な輻輳ウィンドウのチューニングを今後の課題としたい．

謝　　辞

本研究は一部，独立行政法人情報通信研究機構の委託研究「新世代ネットワークを支えるネットワーク仮想化基盤技術の研究開発・課題ウ 新世代ネットワークアプリケーションの研究開発」によるものである．

参 考 文 献

1) android developers:http://developer.android.com
2) Iperf:http://downloads.sourceforge.net/project/iperf/iperf/2.0.4
3) FreeBSD:http://www.freebsd.org/
4) Sourcery G++ Lite 2008q-3-72 for ARM GNU/Linux:http://www. codesourcery.com/, http://www.codesourcery.com/sgpp/lite/arm /portal/release644
5) D.-M. Chiu and R. Jain,“ Analysis of the increase and decrease algorithms for congestion avoidance in computer networks, ” Computer Networks and ISDN Systems, vol. 17, pp. 1-14, 1989.
6) 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．
7) 三木 香央理， 山口 実靖， 小口 正人：無線 LAN 通信環境におけるカーネルモニタを用いた TCP 解析による Android 端末の性能向上手法，DEIM2012，C6-5， シーサイドホテル舞子ビラ神戸，2012 年 3 月．
8) 三木香央理，平井弘実，山口実靖，小口正人: Android 端末の無線 LAN アクセス時の周辺状況に基づく
TCP 制御手法の提案と通信制御ミドルウェアの導入，SACSIS2012，4-6，2012 年 5 月．
9) 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．
10) 平井弘実，三木香央理，山口実靖，小口正人: 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，電子情報通信学会 NS 研究会，NS2011-105，2011 年 11 月