Android端末における通信制御ミドルウェアのタブレット端末への導入と評価

島田　歩実[1]　山口　実靖[2]　小口　正人[1]

概要：近年のスマートフォンやタブレット端末といった無線通信を利用する端末の普及と高性能化に伴い，低帯域かつノイズの多い無線区間における通信性能の向上が求められている．また，近年のロスベースTCP はより高いスループットを確保するためによりアグレッシブな輻輳制御手法を用いているが，有線接続と比べて脆弱な無線接続環境においてはその手法によって膨大なパケットが蓄積され，その結果パケットロスや輻輳などの深刻なエラーが発生してしまうという問題が生じている．先行研究では，ロスベースTCP の一つである TCP CUBIC を輻輳制御アルゴリズムとして使用している Android 端末を用いて，無線 LAN アクセスポイントにおける ACK パケットの蓄積を回避する協調的制御手法がスマートフォン端末向けに提案，実装された．本研究では，スマートフォン端末だけでなく，タブレット端末への導入にも成功し，評価実験の結果よりスマートフォン端末とタブレット端末の振舞の違いを観察した．

Development and Performance Evaluation of Transmission-Control Middleware on an Android Tablet Terminal

Ayumi Shimada[1]　Saneyasu Yamaguchi[2]　Masato Oguchi[1]

1. はじめに

近年のロスベース TCP はより高いスループットを確保するためによりアグレッシブな輻輳制御手法を用いているが，クライアント・サーバ通信において，比較的安定したアクセスポイント (Access Point, 以下 AP) からサーバまでの有線接続区間と比べ，低帯域でノイズによる影響を受けやすく脆弱な，クライアントから AP までの無線接続区間においてはその手法によって膨大なパケットが蓄積され，その結果パケットロスやタイムアウト，輻輳などの深刻なエラーが発生してしまうという問題が生じている． このようなエラーは，無線接続区間において同一 AP に接続するスマートフォン端末の台数が多い場合や端末からの転送量が非常に多大な場合に起きていると考えられている．

そこで，先行研究では，無線区間において同一 AP にスマートフォン端末が多数台接続する場合，AP の通信機会が減ってしまい，AP に膨大なパケットが蓄積されてしまうという問題に着目し，同一 AP につながる複数のスマートフォン端末がお互いの情報を通知し合い，連携した通信制御を行うことで，全ての端末において高速かつ公平な通信を行うことを目的としたミドルウェアの提案と実装が行われてきた [1][2]．端末としては，ロスベース TCP の一つである TCP CUBIC を輻輳制御アルゴリズムとして採用している Android が搭載されたスマートフォン端末が用いられてきた．本研究では，このミドルウェアを Android が搭載されたタブレット端末にも導入し，複数のスマートフォン端末に加えタブレット端末も合わせて実験を行い，その通信性能の検証と評価を行う．

[1] お茶の水女子大学
Ochanomizu University, Bunkyo, Tokyo, 112-8610, JAPAN
[2] 工学院大学
Kogakuin University, Shinjuku, Tokyo, 163-8677, JAPAN

2. 研究背景

2.1 AndroidOS

本研究では，Android プラットホーム上で動作するシステムの開発を行う．Android[1] とは，Google 社によってスマートフォンやタブレットなどの携帯情報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，オープンソースであるため，カスタマイズ性を持つ．

Android はカスタマイズ版 Linux カーネルがベースと

なっており，ロスベース方式の TCP CUBIC[3] を輻輳制御アルゴリズムとして採用している．TCP CUBIC とは，高速ネットワークに適した輻輳制御アルゴリズムであり，TCP BIC[2] のウィンドウサイズ制御を簡素化し，既存の TCP との公平性および RTT(Round Trip Time) 公平性を改善したものである．TCP CUBIC では，TCP の輻輳指標を連続した 2 つのパケット廃棄発生時間の差により，リアルタイムに定義する．TCP CUBIC の増加関数は式 1 で定義される．C はスケールファクタ，t は最後にパケット廃棄が発生した時点からの経過時間，Wmax は最後にパケット廃棄が発生した時点のウィンドウサイズである．

$$W_{cubic} = C(t - \sqrt[3]{\frac{W_{max}\beta}{C}})^3 + W_{max} \quad (1)$$

実際の TCP CUBIC の振舞を図 1 に示す．縦軸は CWND，横軸が時間で，パケットロスを検出したときウィンドウサイズを減少させ，最後のパケット廃棄が発生してからの経過時間をもとに三次関数的にウィンドウサイズを増加させる．

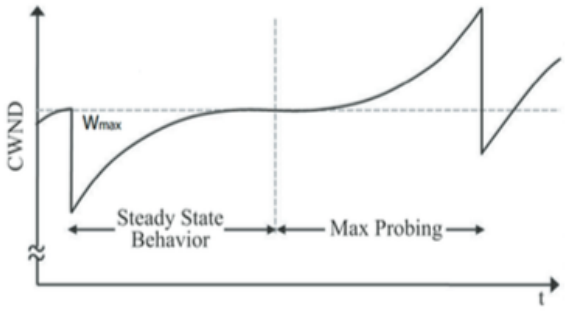

図 1 TCP CUBIC の振舞

また TCP CUBIC はロスベース方式であるため，遅延ベース方式のものと比べて各端末がアグレッシブに通信しすぎる傾向にある．それにより，同時に通信する端末数が多いときには AP で ACK パケットが蓄積しやすく，パケットロスなどのエラーが生じてしまい，その結果エンドホスト側の端末は十分な通信速度を得ることができないという問題点がある．

2.2 カーネルモニタリングツール

カーネル内部の処理は，通常バックグラウンドで進められているため，通常ユーザ空間からその処理の様子を監視することはできない．本研究でベースとして用いられているカーネルモニタは既存研究 [8] により開発されたツールで，Android 端末の通信時における輻輳ウィンドウサイズ (CWND) や往復遅延時間 (RTT) などのカーネル内部の様々なパラメータをリアルタイムにモニタする汎用 PC 向けシステムツールである．図 2 に示したように，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．既存研究 [8] は，すでに Android が搭載されたスマートフォン端末へのカーネルモニタの組み込みに成功しており，本研究においては同様の方法でスマートフォン端末だけでなくタブレット端末への導入も行った．

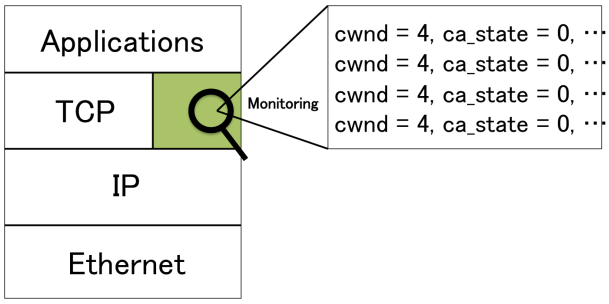

図 2 カーネルモニタリングツール

2.3 輻輳制御ミドルウェア

先行研究 [6][7] で開発された輻輳制御ミドルウェアは，カーネルモニタをベースとしたシステムであり，同一 AP に接続した複数の端末間でお互いの接続状況を把握し，その接続台数によって混み具合を予測し，CWND の上限値を自動で算出し補正する組み込みシステムである．端末間で可用帯域を公平に分け合うことで，無線 LANAP における ACK パケットの蓄積を回避し，複数の端末が同一の AP に接続して通信するときの全体の通信速度と公平性の向上を可能にしている．

具体的な制御としてはまず，UDP パケットを 0.3 秒ごとにブロードキャストすることで，周辺端末に自端末の通信状況を通知する．それと同時に IP アドレスで固有化した周囲からの UDP パケットを受け取り，接続端末数を把握する．その情報と式 2 を用いて CWND の理想値を補正する．また，先行研究 [7] において帯域幅は接続台数に伴い変化することが確認されているのため，帯域幅に関しては，式 3 のように計算した値を用いている．ただし，BWmax はその通信環境における最高通信速度，f(端末数) は端末数の増加による合計通信速度の低下を表す単調減少の関数であり，式 4 を用いている．

$$Ideal_cwnd = \frac{Bandwidth[Mbps] \times RTT[sec]}{Segmentsize(1.5Kbyte) \times Numberofterminals} \quad (2)$$

$$Bandwidth = BWmax \times f(Numberofterminals) \quad (3)$$

$$f(Numberofterminals) = Numberofterminals^{-0.15} \quad (4)$$

これらの式を用いて算出された CWND 値を，プロセスインタフェースに CWND の最大値として 0.5 秒ごとに書き込み補正する．さらに，通信中はカーネルモニタを 0.5 秒単位で常時監視し，RTT とその最小値 (min-rtt) を取得する．min-rtt は，通信中で最も小さい RTT を常に上書きしていくことで値を更新する. 取得した値をもとに式 5 を

用いて RTT の増減の比率 (ratio rtt) を求める．

$$ratio\_rtt = \frac{RTT}{min\_rtt} \quad (5)$$

この ratio rtt が 6.0 以上になると，トラフィックが混んできたとみなし，式 2 を用いた端末数による CWND を補正するフェーズに切り替わる．また，その補正をした上でも RTT が増加してしまう場合においては，ratio rtt が 5.0 以上になると CWND を 1 にして一時的に通信最小モードに移行し，その後に起こり得るさらなる遅延を回避する．

これらの制御によって，通信中においても同じ AP を共有する他端末が通信を始めたことやそれに伴って急に RTT の値が増加したことを本システムが検知すると，リアルタイムに CWND を適切な値に補正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にも均等に帯域を分け合えるよう制御することができる．さらに本システムにおいては，基本的な TCP の輻輳制御アルゴリズムは変更しておらず，これはデフォルト時同様に機能している．ただし，AP 周りの通信状況に基づき，問題が発生した場合にのみ迅速に対応することで，通信の最適化を実現する．

そこで本研究では，近年スマートフォン端末と共に普及しているタブレット端末にもこのシステムを組み込み，従来のスマートフォンに加え，より性能の高い端末が同一 AP に接続して通信を行うときの全体の通信性能の検証実験を行う．

3. 性能評価実験

3.1 実験 1 ：複数台端末の接続実験

図 3 に示すように，サーバ機と AP の間に人工遅延装置 dummynet[4] を挟み，有線部の往復遅延時間を特に輻輳が生じやすい高遅延環境を模擬するために 256ms に設定している．この環境において iperf を用いて通信スループットを測定した [5]．端末には，スマートフォン端末 (Nexus S) とタブレット端末 (Nexus 7) を用い，それぞれの台数を変え，そのときのスループットやカーネルモニタによってモニタされたパラメータ値から，それぞれの端末の振舞を検証，評価した．また，より性能の高いタブレット端末に対しては，遅延をさらに大きくした 1024ms における環境においても同様の実験を行った．

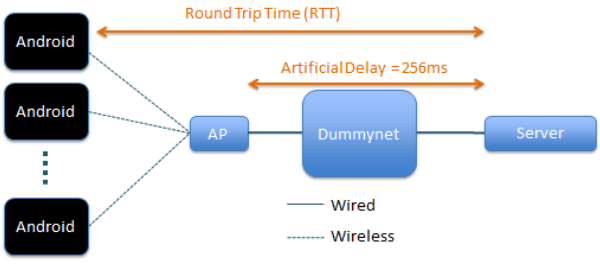

図 3 実験 1 環境

3.2 実験 2 ：複数プロセスの接続実験

図 4 のように，実験 1 と同様の環境下において，一台のタブレット端末を用い，同時に複数のプロセスを動作させた．通信を行うプロセスとして iperf を用い，同時に動作させるプロセス数を変化させ，カーネルモニタで端末の振舞を測定した．

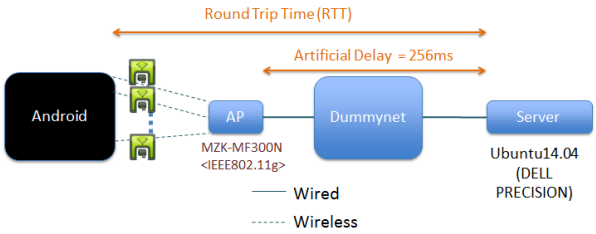

図 4 実験 2 環境

4. 実験結果

4.1 実験 1 ：複数台端末の接続実験結果

図 5 から図 8 に実験 1 の結果を示す．図 5 と図 6 は，端末を同時に 7 台通信させたときの輻輳ウィンドウ (CWND) の振舞を，ミドルウェアによる通信制御を用いた場合とそうでない場合でそれぞれ測定した結果である．図 5 はスマートフォン端末における結果，図 6 はタブレット端末における結果を示している．まず，図 5 に示したスマートフォン端末は，通信制御ミドルウェアがオフの状態で通信した場合は，CWND の上限値が制御されていないため端末が大量のパケットを送出しており，その結果パケットロスなどのエラーが頻繁に起きている様子がわかる．一方，通信制御ミドルウェアがオンの状態で通信をさせた場合は，帯域幅遅延積から自動で算出した CWND の値で制御され，パケット送出量が抑えられていることが確認できる．

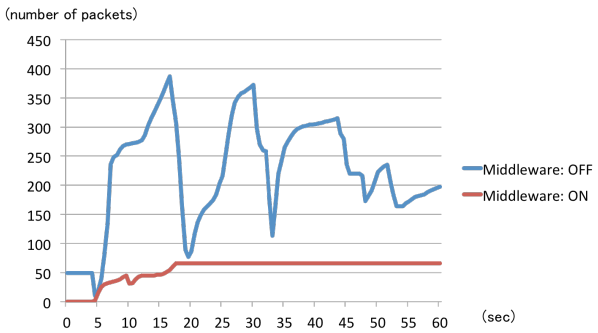

図 5 7 台同時通信におけるスマートフォン端末の CWND の推移

また，図 6 に示したタブレット端末においては，通信制御ミドルウェアをオフにして通信させた場合の CWND は，スマートフォン端末よりも平均して大きく，パケットが大量に送出されている．さらに，エラー発生後すぐに大きい値まで積極的に上がり，再び大量のパケットが送出されている様子がわかる．また，通信制御ミドルウェアをオンに

した場合もオフの場合とほぼ同様の振舞をしており，これは組み込んだミドルウェアはスマートフォン端末向けに開発されているため，タブレット端末にとって適切なコントロールがなされていないためであると予測できる．

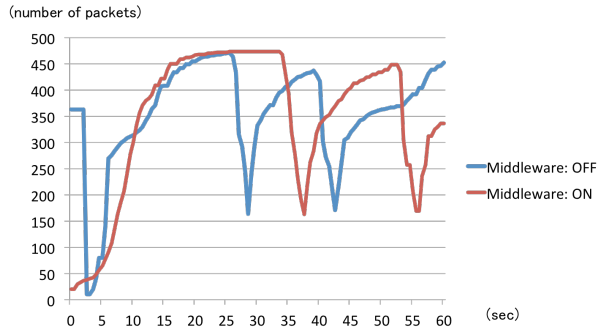

図 6　7 台同時通信におけるタブレット端末の CWND の推移

図 7 と図 8 は，端末数を 1 台通信から 7 台同時通信へ増加させたときのスループットの変化を，通信制御ミドルウェアをオンとオフそれぞれ測定した結果である．図 7 に示したスマートフォン端末に関しては，台数が増加すると全体の通信速度が大きく低下するが，通信制御ミドルウェアをオンにした場合，CWND の正しい補正により，7 台同時通信させた時のスループットの減少を抑えることができていることが確認できる．つまり，通信制御をオンの状態で通信すると，オフの場合と比べて約 3 倍通信速度が向上しており，先行研究 [6] で確認されている結果を再現することができた．

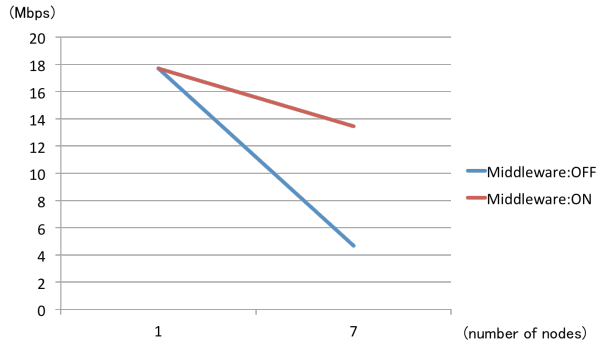

図 7　スマートフォン端末の通信性能の評価（スループット）

図 8 に示すタブレット端末に関しては，図 6 で確認できたように，通信制御ミドルウェアがオンとオフどちらの場合も CWND はほぼ同様の振舞をしているため，スループットにも違いはほぼないことが確認できる．また，エラー発生後にパケット送出を抑えずに，CWND を積極的に大きくしてパケットを再び大量に送出する振舞が全体の通信速度を維持しているため，複数のタブレット端末が一つの AP に接続して通信を行っても 1 台通信のときと比べても通信速度が低下しないと考えられる．

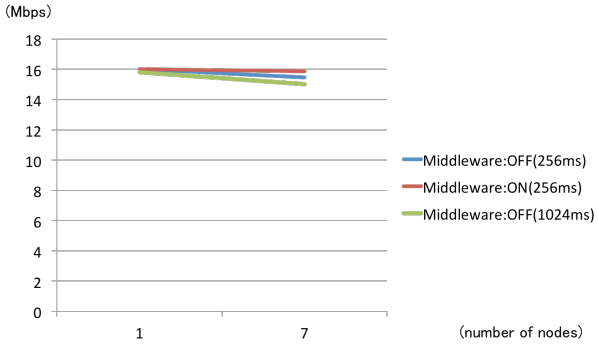

図 8　タブレット端末の通信性能の評価（スループット）

4.2　実験 2：複数プロセス接続実験結果

図 9 と図 10 に，実験 2 の結果を示す．図 8 にはプロセス数増加による CWND の振舞を示す．図 9 を見てわかるように，プロセス数が 1 のときと比べ，複数のプロセスが同時に動作しているときは，CWND の値が大幅に減少していることが確認できる．また，図 10 は RTT の増減を示しているが，プロセス数が 1 のときと比べ，複数のプロセスが同時に動作しているときは，RTT の値が大きくなっていることが確認できる．これらの結果から，CWND や RTT の値を用いて制御行うことで，通信性能が向上する可能性があると考えられる．

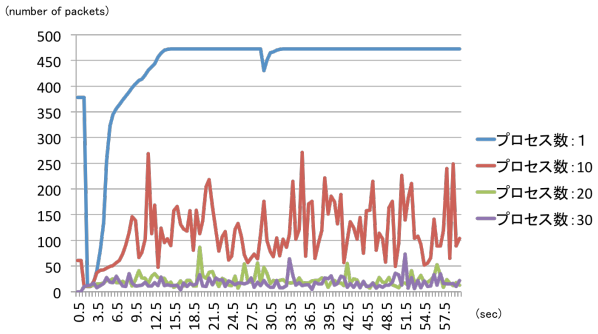

図 9　タブレット内のプロセス数の変化における CWND の推移

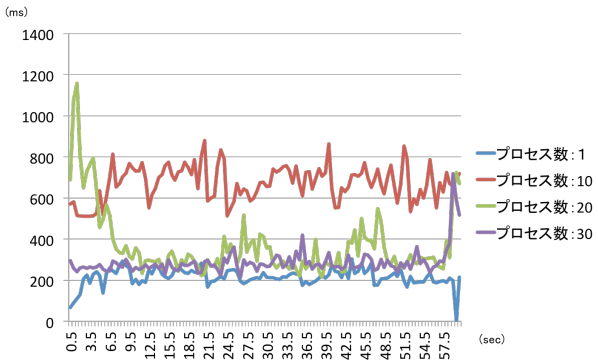

図 10　タブレット内のプロセス数の変化における RTT の推移

5. まとめと今後の課題

本研究では，ロスベース TCP を使用する Android が搭載された携帯端末が多数台同時に通信する場合において，AP に ACK パケットが蓄積し輻輳を起こす問題に着目した．そして，先行研究によって開発された輻輳制御ミドルウェアを Android が搭載されたタブレット端末にも組み込むことに成功し，スマートフォン端末とタブレット端末の通信性能の評価を行った．具体的には，先行研究でスマートフォン端末で行われてきた実験を再現し，さらにタブレット端末においても同様の実験を行い，その振舞から通信性能を評価した．実験の結果，タブレット端末の輻輳ウィンドウの振舞はスマートフォン端末と大きく異なり，スマートフォン端末と比較してよりアグレッシブな通信が行われていることを確認することができた．さらに，タブレット端末は遅延 1024ms の状況において複数の端末を同時通信させてもスループットの低下が見られず，その性能の高さを確認した．しかし，タブレット端末において，輻輳制御ミドルウェアを用いる場合とそうでない場合の輻輳ウィンドウの振舞にほぼ違いが見られないことから，組み込んだドルウェアはスマートフォン端末では適切に CWND をコントロールしているが，タブレット端末では適切にコントロールできていない可能性があると考えられる．

また，タブレット端末の性能の高さに注目し，一台のタブレット端末で複数の通信プロセスを同時に動作させたときの通信の振舞を測定した．実験の結果から，プロセス数が変化すると輻輳ウィンドウの振舞や RTT の大きさが大きく変化することを確認できた．

今後の課題としては，組み込んだミドルウェアはスマートフォン端末向けに実装されたものであり，タブレット端末では適切にコントロールできていない可能性があるため，タブレット端末においても適切に CWND をコントロールするようにミドルウェアを改良する必要があると考えられる．そのためには，輻輳ウィンドウの振舞や RTT の値の変化に注目しながら改良をしていくことが有効であると考えられる．さらに，通信の振舞の異なるスマートフォン端末とタブレット端末が混在して通信を行う場合の全体の通信性能の評価と，その向上のためのミドルウェアの改良が必要であると考えられる． さらに今後は，異なるネットワーク環境での実験を行うため，そのような環境で動作する Nexus 5 への通信制御ミドルウェアの導入と評価実験も検討している．

謝辞

本研究を進めるにあたって，NTT ドコモの早川愛さんより大変有用なアドバイスをいただきました．深く感謝いたします．

参考文献

[1] Android open source project, http://source.android.com
[2] L. Xu, K. Harfoush, and I. Rhee, ”Binary Increase Congestion Control for Fast, Long Distance Networks,” Proceedings of Tech. Report, Computer Science Department, NC State University, 2003.
[3] Sangtae Ha, Injong Rhee and Lisong Xu ”CUBIC: a new TCP-friendly high-speed TCP variant” ACM SIGOPS Operating Systems Review - Research and developments in the Linux kernel, vol.42, pp.64-74, July 2008.
[4] The dummynet project: http://info.iet.unipi.it/ luigi/dummynet
[5] Iperf For Android Project in Distributed Systems, http://www.cs.technion.ac.il/ sakogan/DSL/2011/projects/iperf/index.html
[6] Ai Hayakawa, Saneyasu Yamaguchi, Masato Oguchi: ”Reducing the TCP ACK Packet Backlog at the WLAN Access Point,” Proc. ACM IMCOM2015, 5-4, January 2015.
[7] Hiromi Hirai, Saneyasu Yamaguchi, and Masato Oguchi: ”A Proposal on Cooperative Transmission Control Middleware on a Smartphone in a WLAN Environment,” Proc. IEEE WiMob2013, pp.710-717, October 2013.
[8] Kaori Miki, Saneyasu Yamaguchi, and Masato Oguchi: ”Kernel Monitor of Transport Layer Developed for Android Working on Mobile Phone Terminals,” Proc. ICN2011, pp.297-302, January 2011.