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　（指導教員：小口正人）

1　はじめに

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，無線区間において同一アクセスポイント (Access Point) 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開発が行われてきた．本研究では，このミドルウェアを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を行う．

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 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 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 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．また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うという問題がある．

2.2　カーネルモニタリングツール

本研究でベースとして用いられているカーネルモニタ [3] は既存研究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システムツールである．

2.3　輻輳制御ミドルウェア

先行研究 [1][2] 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 LAN アクセスポイントにおける 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．しかしながら，この先行研究で扱ってきた端末はスマートフォン端末のみであった．

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，よりスペックの高い端末が同一 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

3.1　実験 1

図 1 に示すように，サーバ機と AP の間に人工遅延装置 dummynet 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．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．

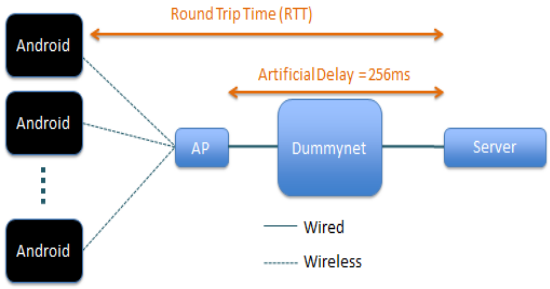

図 1: 実験 1 環境

3.2　実験 2

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．

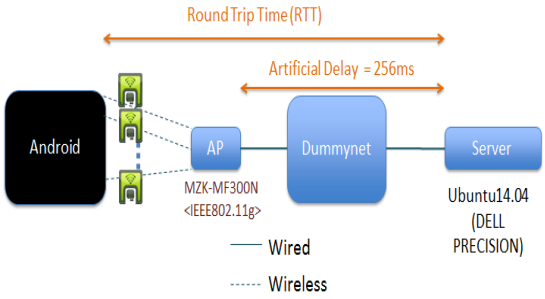

図 2: 実験 2 環境

4　実験結果

4.1　実験 1

図 3，図 4 に実験 1 の結果を示す．図 3 はスマートフォン端末 (Nexus S) とタブレット端末 (Nexus 7) をそれぞれ同時に 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．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．また，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いをしており，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 4 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．スマートフォン端末（図 4 左）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，先行研究で確認されている結果を再現することができた．

タブレット端末 (図 4 右) に関しては，図 3 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い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．

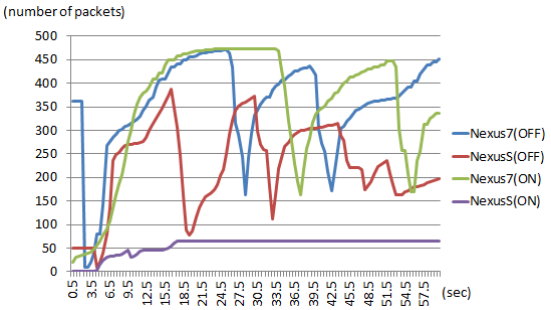

図 3: 7 台同時通信におけるスマートフォン端末とタブレット端末それぞれの CWND の推移

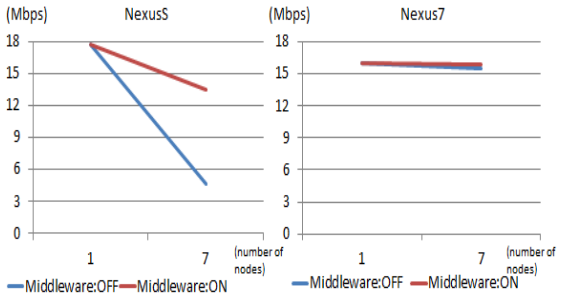

図 4: スマートフォン端末（左）とタブレット端末（右）の通信性能の評価（スループット）

4.2 実験 2

図 5 と図 6 に，実験 2 の結果を示す．図 5 にはプロセス数増加による CWND の振舞いを示す．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 6 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 を制御することで，通信性能が向上する可能性があると考えられる．

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

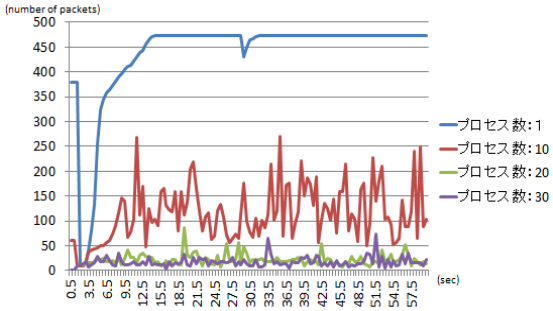

図 5: タブレット内のプロセス数の変化における CWND の推移（左）と RTT の推移（右）

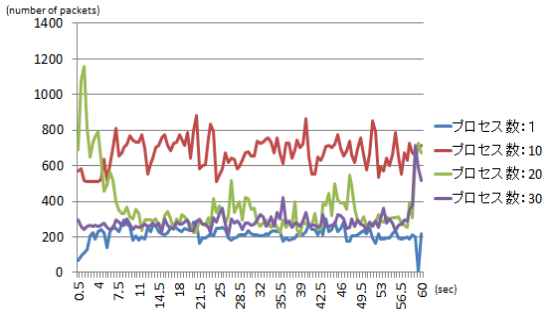

図 6: タブレット内のプロセス数の変化における CWND の推移（左）と RTT の推移（右）

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タブレット端末においても行い，その振舞から通信性能を評価した．また，タブレット端末のスペックの高さに注目し，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，タブレット端末は遅延 256ms 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なかった．また，組み込んだミ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．

今後の課題としては，現在の遅延 256ms においては，タブレット端末は通信性能を保っているが，さらに遅延の大きい状況下においては通信性能が低下する可能性があるため，そのような状況下におけるタブレット端末の通信の振舞を評価する実験を行う必要があると考えられる．また，組み込んだミ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 をコントロールをするようにミドルウェアを改良し，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．

謝辞

本研究を進めるにあたって，工学院大学の山口実靖先生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] Ai Hayakawa, Saneyasu Yamaguchi, Masato Oguchi: ”Reducing the TCP ACK Packet Backlog at the WLAN Access Point,” Proc. ACM IMCOM2015, 5-4, January 2015.

[2] 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.

[3] 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.