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

三木 香央理[†] 平井 弘実[†]
山口 実靖[††] 小口 正人[†]

近年，スマートフォン市場の成長に伴い，携帯端末で動作する組込み機器のソフトウェアプラットフォームとして Android が注目されている．これまでのモバイル端末は通信時に個々に独立した制御を行っており，またそれを改良する研究は多くなされている．最近のモバイル端末，特にスマートフォンは高機能携帯電話とも呼ばれ端末自体の性能も良くなってきているため，単独端末内での閉じた制御ではなく，端末間で通信情報を交換し連携した制御することも可能になってきていると考えられる．本稿では，端末単独で通信の効率化を目指すのではなく，周囲の端末の通信状況を把握することを前提として，TCP のソースコードに手を加えた独自の通信手法を用いて制御を行うことで，他端末との通信状況の兼ね合いを考慮し，帯域を無駄なく有効に使いきることができる通信制御ミドルウェアを提案する．

A Proposal of TCP Control Method based on its Environment for Wireless Access of Android Terminals and Development of Communication Control Middleware

KAORI MIKI,[†] HIROMI HIRAI,[†] SANEYASU YAMAGUCHI[††]
and MASATO OGUCHI[†]

In recent years, with the rapid growth of smart phone market, Android attracts attentions as software platform of embedded system. Previous mobile phones control network communication individually and independently. Some studies have been performed so far improve communication behavior in such a way. It becomes possible that a mobile terminal can exchange information among terminals and control in a cooperative way. Because recent phones, especially smart phones which is called high performance mobile phone, come to have better performance. In this paper, on the premise of such cooperation, we propose middleware which uses original method of communication that considers around condition and uses bandwidth effectively.

1. はじめに

近年，スマートフォン市場の成長に伴い，携帯端末で動作する組込み機器のソフトウェアプラットフォームとして Android[1] が注目されている．これまでモバイル端末において通信時に単独で制御[2]，プロトコルの開発[3] 等に関する研究は多くなされている．しかし，端末が周囲と情報交換し，他の端末の状況を知った上で無線 LAN 通信制御を行おうという発想に基づく研究は行われてこなかった．これは，従来のモバイル端末が処理能力も通信能力も乏しかったため，端末間で制御のための情報交換を行うことがかえってオーバヘッドとなり，性能を損なってしまう可能性が考えられるためである．しかし，モバイル端末，特にスマートフォンは高機能携帯電話とも呼ばれ，処理能力や通信能力が急速に向上しているため，今後のモバイル端末のアクセス環境ではそのような通信制御も有用ではないかと考えられる．そこで本稿では単独端末内の閉じた制御ではなく，これまで研究されていなかった端末間での情報交換を検討し，連携した制御をすることを考える．

本稿では，カーネル内部のパラメータ変化を記録するカーネルモニタツールを Android 端末に実装し[4]，カーネルモニタによって取得した通信状況を周囲の端末間と交換することにより，状況に応じて柔軟に制御する．さらに周囲の状況に応じて TCP のソースコードの輻輳制御部分に手を加えた独自の通信手法[11] に切り替えて通信を行うことで，他端末との通信状況の兼ね

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

合いと公平性を考慮し，帯域を無駄なく有効に使いきることができる新しい通信制御の仕組みを提案する．

本稿は以下のように構成される．まず，2 章でミドルウェアの概念と TCP に手を加えた独自の TCP を用いた提案手法について紹介する．3 章では，実験概要と基礎実験について説明し，4 章では提案手法を用いた制御方法を示す．5 章では，本稿で取り上げた Android 端末 2 機種それぞれで動作する TCP について解析を行い比較する．6 章ではその解析結果を受け，相互に TCP を移植し，公平性の確保を図る．7 章で提案手法を用いたミドルウェアの概要について説明し，その実験について述べる．8 章は本稿のまとめと今後の課題について述べる．

2. 提 案 手 法

2.1 ミドルウェアによる情報交換

モバイル端末の無線 LAN アクセスにおいて，端末間で情報を交換し，周囲の環境に応じてフレキシブルに TCP を切り替えるミドルウェアについて説明する．

本ミドルウェアは，Android 端末が広帯域有線ネットワーク接続されたクラウドサーバと通信する場合を想定し，輻輳が懸念されるアクセスポイント・Android 端末間の無線帯域を共有している他端末の通信状況を考慮した制御を目指している．TCP-CUBIC[5] をベースとする TCP は，通信経路の状況を正確には把握せず，エンド・エンド間のパケット損失に応じて輻輳制御を行っている．しかし，TCP-CUBIC を含むこれまでの TCP に関する既存研究[6][7] は有線志向で開発されたアルゴリズムであり，無線区間と有線区間では大きく性質が異なるため，新たな仕組みが必要と考えられる．

無線区間におけるパケット損失やビットエラーは，ノイズによる影響を受けているため，可用帯域が余っているにも関わらず，これらを輻輳と判断して輻輳ウィンドウを半減させてしまうと，転送効率が悪化する．このようなノイズと実際の輻輳を見分ける手段として Congestion Notification や Loss Notification という Linux の標準オプションが存在する[8]．しかし，この手法は通信経路上のネットワーク機器の対応が必要であるため，導入コストが高く今日においても普及が進んでいない．

そこで本稿では，無線通信を主体とするデバイスの通信において，無線区間の特性を考慮し，同一アクセスポイントを共有する端末間で通信に関する端末情報を通知し合い，無線区間のトラフィック予測を試みる．本手法は様々な無線デバイスに有効と考えられるが，オープンソースで改造が容易であり，またカスタム ROM といった独自の ROM 配布も一般的である Android で実装を行う．

可用帯域を見積もるために端末間で交換する端末情報としては，これから送出するセグメント数を示す輻輳ウィンドウや，ソケットバッファキューの長さといったカーネル内部のパラメータが有効であると考える．このような端末情報を同一アクセスポイントを共有する端末間で通知し合い，混雑具合を判断した上で輻輳制御アルゴリズムの切り替えを行い全ての端末のトータルスループットの向上と公平性の確保を目指す．

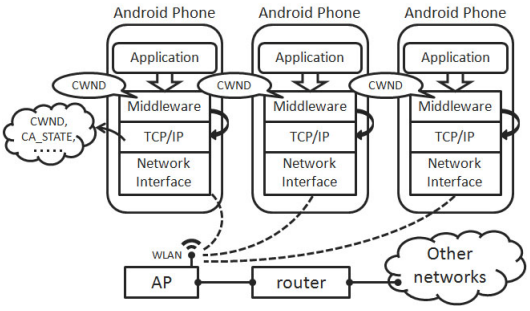

図 1 ミドルウェアの概念図

ミドルウェアを適用させたシステムの例を図 1 に示す．各 Android 端末上のミドルウェアは，各自のトランスポート層を監視し続け，通信が行われると解析によって得られた輻輳ウィンドウサイズを周囲の Android 端末に通知する．通知を受けた他の Android 端末は，輻輳ウィンドウサイズから現在のネットワークの混み具合を予測し，その状況に適した TCP の切り替えを行う．

2.2 独自の TCP 概要

本ミドルウェアは周辺端末の通信状況に基づき，必要な場合にはデフォルトの TCP から動的に切り替えて通信制御を行うが，切り替え用に独自の輻輳制御アルゴリズムを開発した．

独自の TCP は，輻輳ウィンドウの値を利用したパケット送出の積極性の強弱設定により，制御を行う TCP である．既存の輻輳制御アルゴリズムをチューニング及び，輻輳ウィンドウの上限値を設定可能にすることで，このような TCP を実現している．

図 2 の概念図の赤色のグラフに独自の TCP の振舞を示す．デフォルトの TCP は黄緑と水色のグラフに示すようにパケット損失した際に輻輳ウィンドウサイズを大きく減少させているのに対し，独自の TCP は，輻輳が起きた際は輻輳ウィンドウを任意の値まで減少させ，その後輻輳回避フェーズで線形に上昇させ，輻

輻輳ウィンドウを高い値に保つ．通信開始時はスロースタートのフェーズで輻輳ウィンドウを任意の値まで一気に上昇させるところは通常の TCP と同じである．独自のアルゴリズムは高遅延環境において性能向上が確認されている[10]．

また輻輳ウィンドウ上限値の設定は，独自の TCP とデフォルトの TCP の両方の上限値を外部プロセスから調節できるように改造した．

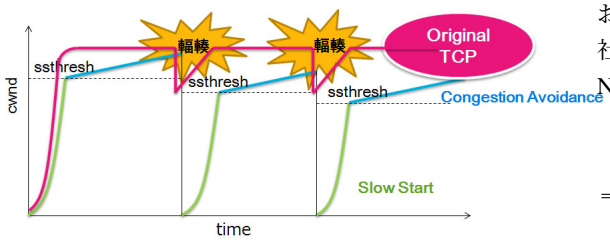

図 2 独自の TCP

3 章で示すようにデフォルトの TCP のみを用いた場合には，Android 端末を用いた基礎実験において，通信のタイミングや周囲の状況によって不必要に輻輳ウィンドウを減少させ不公平な通信状況に陥ることが確認された．周囲の端末の通信状況を把握し，不公平な通信状況に陥っている場合や，可用帯域を使い切れていないと判断した場合の制御手法の 1 つとして独自の TCP で輻輳ウィンドウを高く保つようにする．一方で高性能端末においては通信帯域を占有してしまい，性能の低い端末が帯域を確保することができない場合に，輻輳ウィンドウの上限値を低く抑えることで帯域を分け与える制御を行う．

本稿では，周辺状況に基づく TCP 制御手法として，周囲の端末の状況に応じ，輻輳制御を行う TCP を切り替えたり輻輳ウィンドウの上限値を変えたりする制御手法をミドルウェアで実現する事を考えている．どのように制御すればスループットと公平性を上げる事ができるか，まずは特定の端末を特定数用いて 4 章から 6 章に示すように手動で切り替えて実験を行い，その実現可能性を探る．

具体的には，3 章でデフォルトの TCP を利用した場合の通信スループットを測定し，4 章では，それぞれの環境において，独自のアルゴリズムを用いて輻輳ウィンドウを高く保つ手法と輻輳ウィンドウの上限値を設定し送信量を抑える手法を用いて，各端末が公平性と通信スループットの向上を目指して最適となる制御方法を示す．5 章では不公平が生じる原因を考察し，6 章では 4 章での制御を発展させ，さらに TCP の移植を行うことで，より公平な通信を試みる．7 章では，汎用化に向けて，周辺端末情報を元に実環境のトラフィック混雑具合をミドルウェアが判断し，通信制御を自動的に最適化するミドルウェアを開発し，その動作結果を示す．

3. 実験システムと基本性能測定

3.1 実 験 環 境

本章では本実験で使用した測定ツール，実験環境および実験手順を示す．Android 端末としては，HTC 社製の HT-03A 端末 (以下機種 A) と Samsung 社製の Nexus S 端末 (以下機種 B) を用いて基礎実験を行う．

表 1 Experimental Environment

機種 A	Model number	HT-03A
	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 ERE27
機種 B	Model number	Nexus S
	Firmware version	2.3.4
	Baseband version	I9023XXKD1
	Kernel version	2.6.35.7-kaori1198-ge382d80-dirty
	Build number	GRJ22
サーバ端末	OS	Fedora release 10 (Cambridge)
	CPU	Intel(R) Pentium(R) 4 CPU 3.00GHz
	Main Memory	1GB

表 1 に本稿の実験環境を示す．本稿では，スループット測定のためにクロスコンパイルした iperf-2.0.4[12] の実行ファイルを Android 端末に転送し，ソケット通信の性能を測定した．クロスコンパイラとして arm-2008q3[13] を使用した．

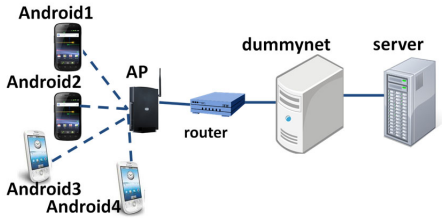

図 3 実験機器構成図

まず，図 3 に示すように IEEE802.11g で 2 種類の Android 端末計 4 台が無線 LAN アクセスポイントを経由し，サーバへパケット転送を行った場合の基本性能を測定した．図 5 に実験結果を示す．

3.2 基本性能測定

図 4 に改造を行っていない端末で通信を行った時の

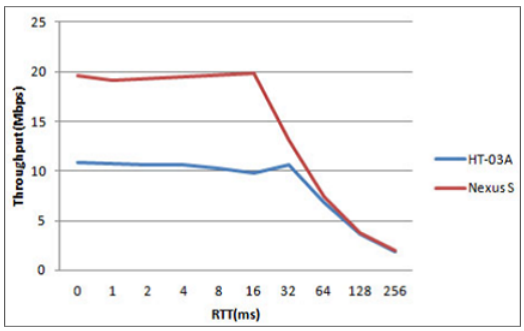

図 4 各機種が単独で通信を行った時の通信性能

実験結果を示す. 機種 A は 2009 年始めに発売された端末であるのに対し，機種 B は 2010 年末に発売された端末である．近年，Android 端末の成長は大きく，機種 A と機種 B はカーネルアップデートに伴う TCP の改良だけでなく，端末本体のハードウェアの処理能力においても大きな差が生じている．往復遅延 16ms 以下における通信スループットは，機種 A が約 10Mbps，機種 B が約 20Mbps であることから，機種 A と機種 B の通信性能差は 1:2 と考える．つまり，これらの機種が混在して同時に通信を行った場合にも 1:2 の性能差になることが望ましいが，実際に機種 A を 2 台，機種 B を 2 台でアクセスポイントを共有し，合計 4 台の端末で同時に通信を行ったところ，図 5 に示す結果となった．図 4 では，低遅延環境におけるトータルスループットは，802.11g の最高転送速度である 54Mbps に近づくどころか，機種 B の単独通信速度である約 20Mbps にすら達しておらず，可用帯域を大きく余らせていることがわかる．また公平性においては，単独通信時の性能差である 1:2 を考慮しても，機種 A の通信速度が特に大きく劣化していると言える．

4. 特定環境における TCP 制御手法

3.2 節の結果を受け，環境別に通信性能向上を目指したチューニングを行った．制御手法を表 2 に示す.

表 2 TCP 制御方法

RTT	機種 B		機種 A	
	TCP	上限値	TCP	上限値
0	default	20	original	70
1	default	20	original	70
2	default	20	original	70
4	default	20	original	70
8	default	20	original	70
16	default	20	original	100
32	default	30	original	100
64	default	55	original	100
128	default	55	original	100
256	default	100	original	100

TCP の default とは，OS に標準で実装されている TCP を用いる場合である．TCP の original とは，輻輳ウィンドウを増加しやすく，減少しにくいようにカーネルを改造し，アルゴリズムの積極性を強めた TCP を指す．また上限値は 3.2 節に示したように帯域を占有してしまう場合に輻輳ウィンドウ値が過度に増加することのないように default と original の双方に実装したものである．表 2 に示す値は本実験における通信スループットの向上が最も高く見られた最適値である．競合時に，機種 B はハードウェアの処理能力も高く，OS の通信制御においても標準で積極性が強いため，輻輳ウィンドウの上限値を抑えることで帯域を取りすぎないように制御する．一方で，機種 A の標準 TCP は保守的な振る舞いをするため帯域を確保することが難しいので，積極性が強い独自の TCP に切り替える．

図 6 に表 2 の制御手法にしたがって通信実験を行った時の実験結果を示す. 図 5 と比較して，全ての端末において通信性能が向上することが確認された．2 種類の機種が混在する競合環境で通信を行った場合，機種 A よりも高性能な機種 B は，パケット転送量を意

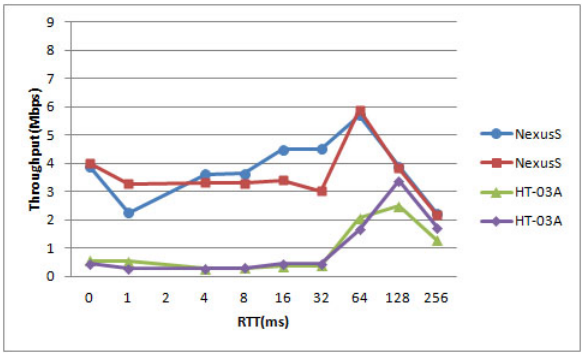

図 5 4 台の端末が同時に通信を行った時の各端末の通信性能

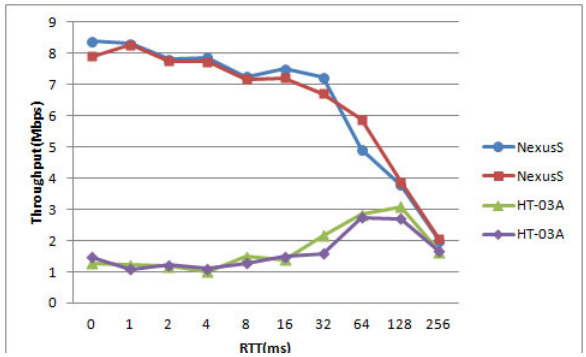

図 6 4 台の端末が表 2 の制御に従って通信を行った時の通信性能

図的に抑える方がむしろ自身の性能が向上することが確認された．しかし，機種 A の通信スループットは標準 TCP の 2 倍に向上したものの不公平が改善されていない．

このような制御を行っても通信速度が十分向上しない原因として機種 A と機種 B の標準 TCP に大きな違いがあることが挙げられる．大きな特徴の違いとして，機種 A の標準 TCP は有線志向で開発された TCP-CUBIC を引き継いだ保守的な TCP であり，輻輳ウィンドウを減少させやすい．機種 B は無線環境でのノイズによる影響を考慮し，多少のビットエラーでは異常状態に陥らず輻輳ウィンドウをほとんど減少させないように設計されていると考えられる．次章では各端末に標準で実装されている TCP の振舞いの違いを解析する．

5. 同機種による振舞の解析

まず，図 3 に示すものと同じ実験環境において，4 台とも全て機種 A を用いた場合，4 台とも全て機種 B を用いた場合で，標準 TCP を使って通信した場合の輻輳ウィンドウサイズと CA_STATE（後述）を解析した．

5.1 同機種 4 台同時通信時の輻輳ウィンドウサイズ

図 7，8 に機種 A と機種 B をそれぞれ 4 台ずつ用いて通信を行った際の輻輳ウィンドウの遷移を示す．

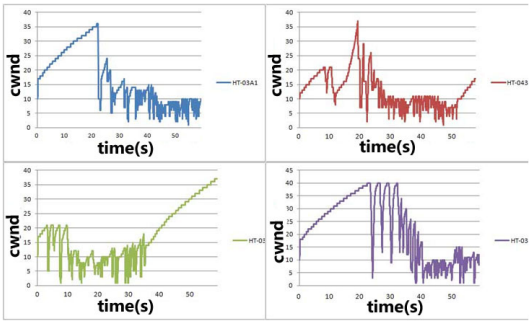

図 7 機種 A の 4 台同時通信を行った時の輻輳ウィンドウサイズ

図 7，8 から，機種 A は細かい変化で逐一輻輳ウィンドウサイズを増減させて調節しているのに対し，機種 B は輻輳ウィンドウサイズを安定して大きな値に保ち，ほとんど減少させていないことが分かる．

5.2 同機種 4 台同時通信を行った時の CA_STATE

次に図 9，10 に機種 A と機種 B をそれぞれ 4 台ずつ用いて通信を行った際の CA_STATE を示す．CA_STATE とはカーネルのソースコード内で用いられるパラメータで TCP の状態を表し，0 は正常状態を示す Open で，1 から 4 はパケット順序の入替わり検出の Disorder，輻輳通知の CWR，高速再転送等の Recovery，タイムアウトで遷移する Loss といった異常状態であり，番号が大きいほど状態が酷い異常状態を示す．

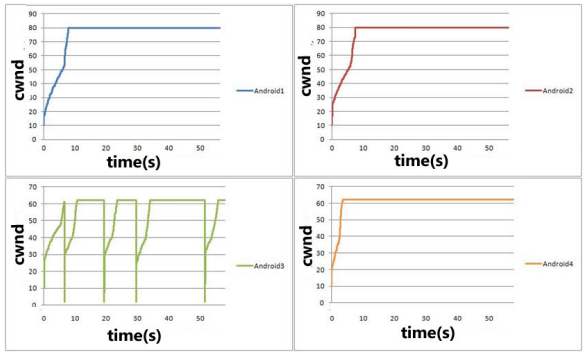

図 8 機種 B の 4 台同時通信を行った時の輻輳ウィンドウサイズ

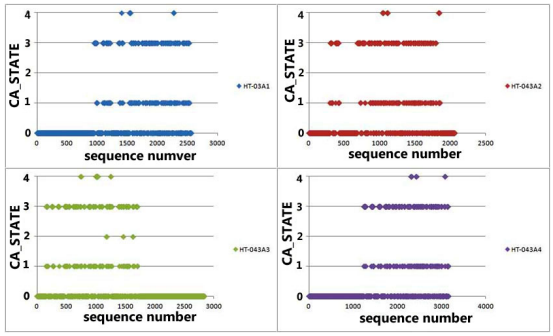

図 9 機種 A の 4 台同時通信を行った時の CA_STATE

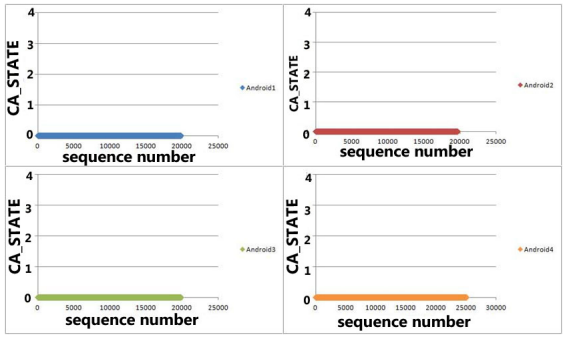

図 10 機種 B の 4 台同時通信を行った時の CA_STATE

図 9，10 から機種 A は異常状態に頻繁に陥っているのに対し，機種 B は全く異常状態に陥っていないことが分かる．

本章の結果とそれぞれのソースコードを比較した結

果，機種 A では異常状態に遷移していた箇所を機種 B では異常状態に遷移しないようにしていることが多いことがわかった．具体的には，機種 B の標準 TCP はヘッダファイルが大きく書き換えられ，新しい関数や変数が多数追加され，3 ウェイハンドシェイクの効率化や高速再転送時にタイマをリセットし輻輳ウィンドウを減少させない，また再転送時も再送スタンプを初期化するなど，機種 A に比べパケット送出量を減少させにくい実装になっているといえる．

以上の実験から，それぞれ同機種間で公平性はとれているが，機種 A と機種 B の混在する環境の場合，標準 TCP では TCP 仕様が違うため公平性が保てず，性能が低下してしまうことが分かった．

6. TCP の移植による公平性の確保

他機種混在環境で機種 A の性能が極端に低下してしまうのは，機種 A と機種 B の両者の標準 TCP で公平性がとれていないことが原因に挙げられる．そこで本章では機種 A の標準 TCP 制御を機種 B に移植し，公平性を保つことが可能であるか調べ，性能に与える影響を検討した．

6.1 機種 B に機種 A の標準 TCP を移植

機種 B は積極性の強い TCP を持ち，単独通信では高い性能の通信を行える．しかし，基礎実験 3.2 節，4 章に示すように，種類の違う TCP と帯域を分け合って通信を行う場合，提案手法を用いて機種 B のパケット送出量を減らすと性能向上が確認されたことから，機種 B の標準 TCP は積極性が強くパケットを送信し過ぎて性能低下を引き起こしていたことが分かる．

一方，機種 A の標準 TCP は保守的で，異常状態に陥りやすく，機種 B と競合した場合，機種 B に帯域を奪われてしまうため性能を上げることができない．ここでは保守的な通信制御を行う機種 A が帯域を確保できるように，機種 B に機種 A の標準 TCP を移植し，必要に応じて切り替えられるようにすることで双方の公平性確保を図る．具体的には機種 B の標準 TCP では正常状態とみなす制御も機種 A と同じように保守的な処理を行うようにした．この移植した TCP に切替え可能な独自のカーネルを用いて基礎実験 3.2 節の図 3 と同じ環境で実験を行う．

6.2 TCP 移植と制御手法

さらに提案手法にある輻輳ウィンドウサイズの上限値制御を機種 A の標準を移植した機種 B に適応する．各環境ごとに最適化した制御方法を表 3 に示す．

ここで機種 B では移植した TCP を用いて通信を行い，さらに帯域を取り過ぎてしまわないように輻輳ウィンドウの上限値を抑えるように制御した．機種 A は輻輳ウィンドウの上限値だけを設定した標準の輻輳制御アルゴリズムを用いて実験を行った．実験結果を図 11 に示す．

表 3 TCP 制御方法 2

RTT	機種 B		機種 A	
	TCP	上限値	TCP	上限値
0	機種 A の TCP	30	default	70
1	機種 A の TCP	30	default	70
2	機種 A の TCP	35	default	70
4	機種 A の TCP	35	default	70
8	機種 A の TCP	35	default	70
16	機種 A の TCP	40	default	70
32	機種 A の TCP	50	default	70
64	機種 A の TCP	65	default	70
128	機種 A の TCP	80	default	70
256	機種 A の TCP	80	default	70

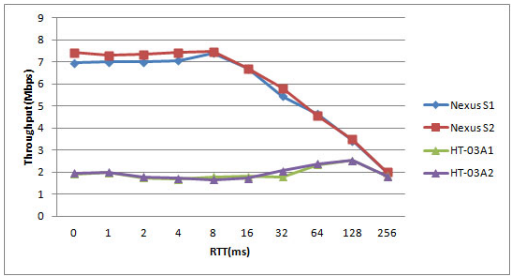

図 11 4 台の Android 端末が表 3 の制御手法に従って通信を行った時の通信性能

図 11 から機種 B に保守的な TCP を移植し，輻輳ウィンドウサイズの上限値を制御することで，通信性能が安定し，機種 A も通信性能を向上させることができたことが分かる．前述したように 4 台で通信を行った際の理想的な通信性能比は，ハードウェアと OS の性能比として機種 A:機種 B は 1:2 であることから，スループットもこの割合で帯域を割った値である約 3.5(Mbps):7(Mbps) 程度である．

TCP の移植と輻輳ウィンドウ上限値の制御を合わせたところ，機種 B に関しては理想的な値で通信を行え，機種 A に関しても性能向上が確認された．

7. 提案手法の切り替え機能を実装したミドルウェア

7.1 通信制御ミドルウェアの実装

前章までの議論を踏まえ，機種 B において機種 A から移植した TCP と OS の標準 TCP を proc インタフェースから切り替えるミドルウェアを開発した．Android におけるソフトウェア開発は，Dalvik VM を利

用した Android アプリケーションとして実装することが一般的であるが，ミドルウェアとして他の通信アプリケーションと共存させるには負荷が大きすぎるため，Android の C ライブラリである bionic を利用したネイティブコードによる実装を行った．

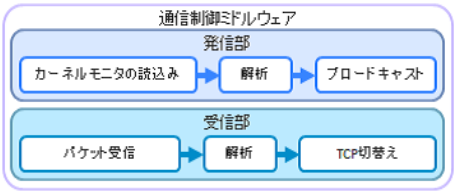

図 12　ミドルウェアの構成

本ミドルウェアは発信部と受信部に分かれて，独立した 2 つの実行ファイルとして機能する．図 12 にミドルウェアの構成を示す．発信部は 0.5 秒間隔でカーネルモニタのログを読み込み，ログがあれば解析して輻輳ウィンドウの値を UDP パケットでブロードキャストする．

一方で受信部はこの通知パケットを受信し，その通知パケットの送信元が機種 A であれば，機種 A が通信を始めたと判断し，機種 B の TCP を切り替え，輻輳ウィンドウの上限値を設定する．また，他端末の発信した通知パケットを一定時間受信しないときに積極性が強い標準 TCP に切り替える．ある時の制御例を図 13 に示す．水色と黄色のグラフが機種 B の輻輳ウィンドウの遷移であり，桃色と緑色のグラフが機種 A の輻輳ウィンドウの遷移である．始めに 50 秒間の通信を開始した機種 B は，他 3 台の端末が途中から通信を開始したことに気づき，TCP を切り替える．また他 3 台の端末が通信を完了させたことに気づき，TCP を標準に戻して積極的な通信を行っている．

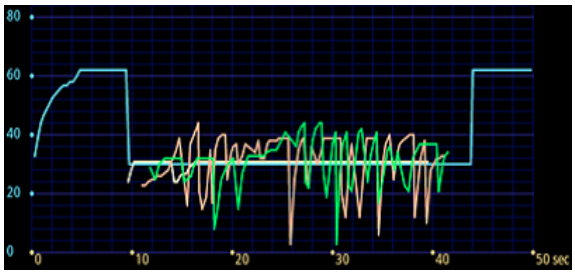

図 13　ミドルウェアの制御例

7.2　通信制御ミドルウェアを導入した通信実験

Android 実機における TCP の切り替えを確認するため，本ミドルウェアを導入した機種 A と機種 B を用いて，RTT=8ms における通信実験を行った．本実験モデルを図 14 に示す．発信部からは通信中の輻輳ウィンドウサイズが通知パケットとして送信され，ミドルウェア受信部により受信される．

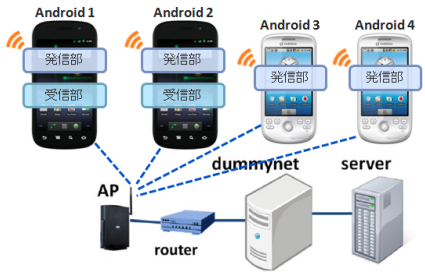

図 14　ミドルウェアの制御を確認する実験モデル

以下の実験例においては，1 台の機種 A 端末が 50 秒間の通信を始め 10 秒が経過した時点で，残りの 3 台の Android 端末が 30 秒間の通信を開始する．3 台の Android 端末が通信を終えると 50 秒間の通信を行っている機種 A 端末は，1 台で帯域を独占して残りの 10 秒間の通信を行う．

実験 1 では比較実験としてミドルウェアの TCP 切り替え機能を適用せず，常時それぞれの標準 TCP で通信をした．実験 2 ではミドルウェアを適用し，TCP を自動的に切り替える処理を行った．帯域を独占したと判断する一定時間は 3 秒間とし，他端末からの通知パケットが 3 秒間届かなかったら提案手法の TCP から標準 TCP に切り替えるようにした．

7.3　実験結果と考察

TCP の振舞は，毎回同じ結果を得られるわけではないが，これらの実験 1，2 の典型的な振舞を図 15，16 に示す．これらのグラフは 4 台の Android 端末で通信実験を行った際の各端末の輻輳ウィンドウの時系列変化である．独自のアプリケーションで，各端末のミドルウェア発信部からブロードキャストされた輻輳ウィンドウの通知パケットを受信し，時系列に従って可視化を行った．縦軸は輻輳ウィンドウサイズ，横軸は時間 (s) を示す．

図 15，16 においては，水色と黄色のグラフが機種 A の輻輳ウィンドウの振舞を示し，桃色と緑色のグラフが機種 B の振舞を示す．実験 1 では，2 台の機種 B が輻輳ウィンドウを大きく増加させ，一方で機種 A はなかなか輻輳ウィンドウを上げられず控え目の制御となっている．同じ帯域を共有していても，機種 B は機種 A に比べて異常状態に陥りにくいため，機種 B は積極性が強い TCP で通信を続けた結果，機種 A の異

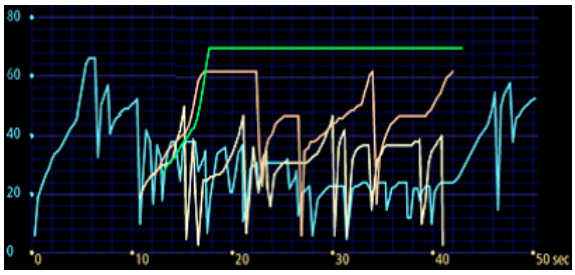

図 15 実験 1: ミドルウェアを適応しない場合の輻輳ウィンドウ時系列変化

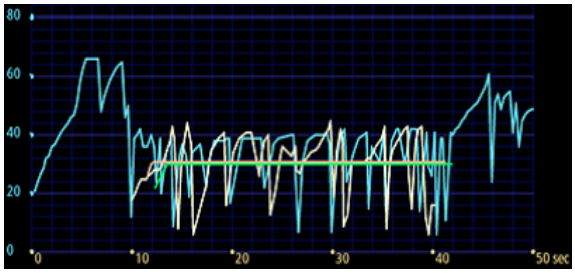

図 16 実験 2: ミドルウェアを適応した場合の輻輳ウィンドウ時系列変化

常状態が増え，輻輳ウィンドウを増加させにくくなったと考えられる．よって通信スループットも低くなってしまう．実験 2 では，機種 B のミドルウェアが他端末の存在を確認した際に，輻輳ウィンドウの上限値を抑えているため，機種 A は残された帯域を積極的に利用することが可能となり，実験 1 よりも輻輳ウィンドウの遷移が明らかに安定したことがわかる．

このように通信制御ミドルウェアを導入したことにより，競合時に自動的に TCP 及び輻輳ウィンドウの上限値を切り替えることにより，標準 TCP の処理よりも安定した輻輳制御処理が可能となった．

8. ま と め

本稿では，Android 実機に適用したカーネルモニタツールを用い，Android 端末の通信時における輻輳ウィンドウやカーネルの状態を解析した．その結果，複数台の Android 端末を通信させた際，効率良くネットワーク帯域が使用されていないことを確認した．そのような場合，端末側の TCP のソースコードに手を加え制御する手法を提案し，更に TCP の公平性を考慮し TCP の移植を行った．また，これらの結果に基づき，TCP の通信処理中に周囲の環境に合わせて自動的に本制御方法の TCP を切り替えるミドルウェアを開発した．本手法は公平性も考慮しているため，ネットワーク帯域を無駄なく使用し，周囲の全端末の通信性能を向上させることができ，有用であるといえる．

本ミドルウェアの実装はまだ初期段階であり，以下のような課題が挙げられる．現在は，環境別に最適制御パラメータを蓄積する方法で，環境に適応することを可能にいているが，今後はより多くの環境に適応させるため，ミドルウェアが周辺端末情報から，可用帯域を見積もった上で，その環境に見合った制御を行うことを目指す．また本手法は，本ミドルウェアを実装した端末間の制御を想定している．ミドルウェアを実装していない端末が混ざった場合には，その端末は独自の判断でパケット送信を行うため，そのような端末と同じアクセスポイントで同居することは可能であるものの，本手法による制御効果は薄れる．また本ミドルウェアにより情報交換を行う端末間の認証やセキュリティ等も課題である．今後はこれらの課題に取り組みながら，汎用性の高いシステムの実現を目指したい．

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．

また本研究を進めるにあたり，ご支援して下さった株式会社 KDDI 研究所の竹森敬祐さん，磯原隆将さんに深く感謝致します．

参 考 文 献

1) Android:http://www.google.co.jp/mobile/android
2) 塩田 尚基，富森 博幸，奥山 嘉昭，浅井 伸一，佐藤 直樹，”通信ポリシを利用したマルチベアラ利用制御ミドルウェア” 情報処理学会研究報告. MBL, [モバイルコンピューティングとユビキタス通信研究会研究報告]，09196072，一般社団法人情報処理学会，no.44，pp7-12，May 2007.
3) 坪井 祐樹，相田 仁，”無線環境における複数経路通信プロトコルの検討”，情報処理学会研究報告. EIP，[電子化知的財産・社会基盤]，09196072，一般社団法人情報処理学会，no.11，pp1-7，September 2011.
4) Kaori Miki, Saneyasu Yamaguchi, and Masato Oguchi: ”Kernel Monitor of Transport Layer Developed for Android Working on Mobile Phone Terminals,” In Proc. the Tenth International Conference on Networks (ICN2011), pp.297-302, St. Maarten, The Netherlands Antilles, January 2011.
5) Sangtae Ha, Injong Rhee, and Lisong Xu, ”CUBIC: A New TCP-Friendly High-Speed TCP Variant”, ACM SIGOPS Operating Systems Review, Volume 42 Issue 5, pp.64-74, July 2008.
6) Luigi A. Grieco Politecnico di Bari, and Saverio Mascolo, ”Performance evaluation and comparison

of Westwood+, New Reno, and Vegas TCP congestion control”, ACM SIGCOMM Computer Communication Review, Volume 34 Issue 2, April 2004.
7) Habibullah Jamal and Kiran Sultan, ”Performance Analysis of TCP Congestion Control Algorithms”, International Journal of Computers and Communications, Issue 1, Volume 2, pp.30-38, 2008.
8) Sally Floyd, ”TCP and Explicit Congestion Notification”, ACM SIGCOMM Computer Communication Review, 1994.
9) Hasegawa Go, Murata Masayuki, ”Transport-layer protocols for high-speed and log-delay networks” The Institute of Electronics, Enformation and Communication Engineers, Technical Committee Conferences,February 2007.
10) 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” マルチメディア, 分散, 協調とモバイル (DICOMO2011) シンポジウム, 7H-2, pp.1407-1414, 2011 年 7 月
11) 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
12) Iperf:http://downloads.sourceforge.net/project/iperf/iperf/2.0.4
13) Sourcery G++ Lite 2008q-3-72 for ARM GNU/Linux:http://www.codesourcery.com/, http://www.codesourcery.com/sgpp/lite/arm/portal/release644