Android端末における遅延時間を考慮したミドルウェアの高機能化

早川　愛† 平井　弘実† 山口　実靖†† 小口　正人†

† お茶の水女子大学　〒 112-8610 東京都文京区大塚 2-1-1
†† 工学院大学　〒 163-8677　新宿区西新宿 1-24-2
E-mail: †{ai,hiromi}@ogl.is.ocha.ac.jp, ††sane@cc.kogakuin.ac.jp, †††oguchi@computer.org

あらまし　近年スマートフォンの爆発的な普及と高機能化に伴い，その大容量高速通信に対する需要が高まっている．しかしながらスマートフォンのようなモバイル端末は，低帯域かつノイズの影響を受けやすい無線通信を利用する機会が多いため，その通信性能を向上することが求められている．そこで，スマートフォンの中でも最もシェア率が高くオープンソースソフトウェアを用いている Android 端末を対象にし，同一アクセスポイントにつながる端末がお互いの情報を通知し合い，連携した通信制御をすることで，全ての端末において高速かつ公平な通信をすることを目的としたミドルウェアの開発が行われている．本研究では，往復遅延時間 (RTT) が通信速度にもたらす影響を明らかにし，この情報を用いてミドルウェアの改良を目指した検討を行う．

キーワード　Android，無線通信，往復遅延時間，輻輳制御

An implementation of Transmission-Control Middleware by Round Trip Time on Android Terminal

Ai HAYAKAWA†, Hiromi HIRAI†, Saneyasu YAMAGUCHI††, and Masato OGUCHI†

† Ochanomizu University　2-1-1 Otsuka, Bunkyou-ku, Tokyo, 112-8610, JAPAN
†† Kogakuin University　1-24-2 Nishi-shinjuku, Shinjuku-ku, Tokyo,163-8677, Japan
E-mail: †{ai,hiromi}@ogl.is.ocha.ac.jp, ††sane@cc.kogakuin.ac.jp, †††oguchi@computer.org

Key words　Android, Wireless LAN, Round Trip Time, Congestion control

1. はじめに

近年，スマートフォンの高機能化が進んでいる．それに伴い，新機種が次々と市場に出回りスマートフォンが爆発的に増加している．また従来の携帯電話は電話と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して機能し，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，自分の気に入ったメーラやブラウザの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．

ネットワークを利用するアプリケーションは，Twitter や Facebook のようにサーバと連携して情報を受発信するものと，Skype や LINE のようにクライアント同士で情報を受発信するものに分けることができる．スマートフォンは小型コンピュータという位置付けではあるものの，前者のように膨大なデータ管理が必要なアプリケーションにおいては，リソース不足で処理しきれないため，クラウドサーバ上にそのデータを保持し，大容量通信を行うことで対処している．スマートフォンを利用する上ではこの通信性能が極めて重要であると言えるため，本研究では，クライアント・サーバ間通信におけるクライアント側からの発信におけるパケット転送制御に注目した．

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．モバイル端末が発信するデータ量のみでは広帯域な有線通信経路上でバッファ溢れを起こす可能性は低いと考えられる．無線空間では，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，トラフィックにも変動がある．すなわち発生するパケットロスの大部分は同じアクセスポイントを共有する端末が多い時や一人一人の転送量が多大な時に無線通信区間で起きていると考えることができる．

これまでモバイル端末において通信時に単独で制御 [1]，プロトコルの開発 [2] 等に関する研究は多くなされているが，本研究が注目する既存研究 [3] においては，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発信の際に，クライアントのアクセスポイント周りで，互いの通信状況を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うという方針の通信制御システムの開発が行われている．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，現在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格段に向上したため，このような他端末と連携した制御が可能になっている．

このシステムでは，制御パラメータとして同一アクセスポイントを共有する周辺のアクティブな通信端末数が用いられている．本研究ではより通信性能を向上させるための別の制御パラメータとして往復遅延時間 (RTT) に着目し，それが通信性能に及ぼす影響を明らかにした．さらに，その影響を考慮した制御をすることでこのシステムのさらなる高機能化へとつなげる．

2. Android OS

Android は，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り，Google 社を中心として開発が行われている．また，2012 年第 3 四半期では全世界のスマートフォン OS の中でも，72.4%とトップシェアを占めている [4].

図 1 に示すように，Android は Linux 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 [5]．Linux OS と大きく異なる部分は，独自に開発された Android 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であるため，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．

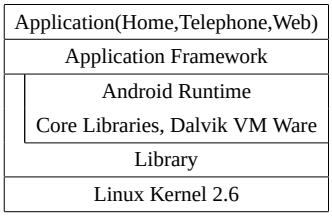

図 1　Android のアーキテクチャ

2.1 Android アプリケーション

Android は，無償で提供される開発環境において構築することができ，オープンソースである点からも対応アプリケーションが開発しやすく数も増えるというメリットがある．また Android はキャリア間の制約がないため，アプリケーション開発においても自由度及び汎用性が高いだけでなく，一度マーケットに登録すると，世界中の Android ユーザからインストールが可能となる．現在 Android マーケットでは，このような大きなビジネスチャンスを提供されているため，毎年多くのアプリケーションが登録されており，アプリケーション市場は賑わっている．

Android マーケットの存在により，ユーザから見てもアプリケーションの入手は容易である．Dalvik 実行形式のバイトコードの状態で配布されているため，必要なアプリケーションをインストールして，スマートフォンを自由にカスタマイズできる．広告から収益を得ることによりアプリケーション自体は無償で提供されているものも多く，気軽にインストールして利用できる．

本研究はこれらのサービスを提供するシステムプラットフォームとしての Android に焦点を当て，通信システムの高速化を目指しているが，このように Android 端末においてアプリケーションの存在を無視することはできない．そこで本研究ではアプリケーションからの無線通信利用を前提として，通信スループットの高速化を目指す．

3. 既存研究

3.1 通信制御ミドルウェア

本研究で改良を加える通信制御システムの概要を説明する．このシステムでは，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．そこで図 2 に示すように，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通知し合い，周辺のアクティブな通信端末数を把握することで，全体のトラフィックを予測し，周囲の他端末の通信状況に応じて，輻輳制御アルゴリズムを適切に補正する．それにより，単独に通信するよりも精密な帯域見積りが実現可能となる．

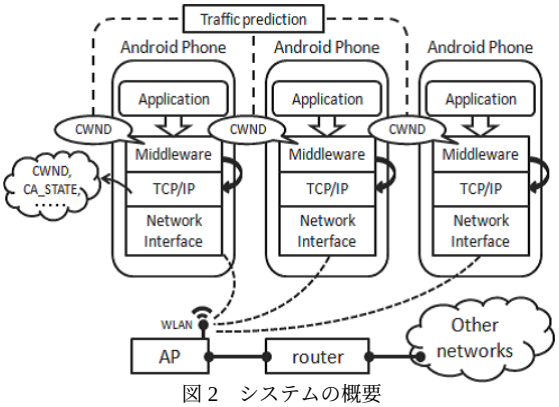

図 2　システムの概要

3.2 カーネルモニタ

前項で述べたシステムのベースとして用いられているツール (図 3) を紹介する．カーネルモニタは，通信時におけるカーネル内部のパラメータ値の変化を記録できる，オリジナルシステムツールである．カーネル内部の TCP ソースにモニタ関数を挿入し再コンパイルすることで，輻輳ウィンドウ値やソケットバッファのキュー長，各種エラーイベントの発生タイミングなどの TCP パラメータを見ることができる [6].

このツールを Android に組み込み，解析を行う．

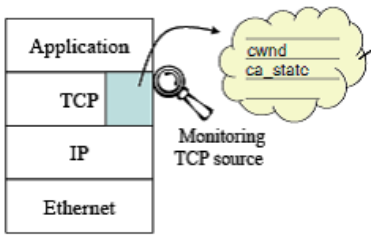

図 3　カーネルモニタ

4.　端末数と通信性能に関する基礎実験

4.1　実 験 概 要

図 4 に示すように，サーバ機と Android 端末の間に実環境を模擬するための人工遅延装置 dummynet を挟み，端末を 1〜10 台通信させた時の各 RTT におけるスループットを Iperf [7] を用いて測定した．さらに，カーネルモニタにて輻輳ウィンドウ値とエラーイベントを取得し，同時に 1 台の端末に対して ping コマンドを用いて実際にかかる RTT の時間変化を調べた．

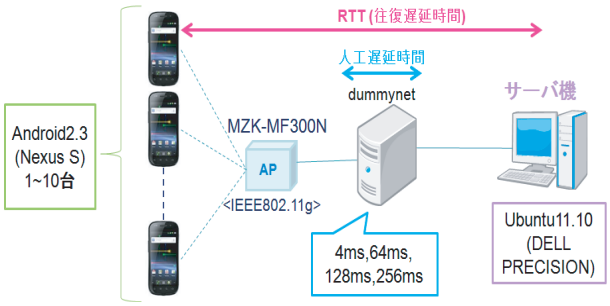

図 4　実験トポロジ

4.2　実 験 環 境

本実験で使用した実験環境を表 1 に示す．また，無線通信方式は IEEE802.11g で行った．

表 1　実 験 環 境

Android	Model number	Nexus S
	Firmware version	2.3.4
	Baseband version	I9023XXKD1
	Kernel version	2.6.35.7-hiromi0824
	Build number	GRJ22
server	OS	Ubuntu 11.10 (64bit) / Linux 3.0.1
	CPU	Intel(R) Core 2Quad CPU Q8400
	Main Memory	7.8GiB

4.3　実験結果と考察

図 5 に通信台数とスループット (平均と合計) の関係を示す．図 5 より，通信端末数を増やすと，合計通信速度が大幅に低下することが分かる．さらに、低下する台数も人工遅延時間によって違うことがわかった．

この原因を調べるために，劣化が見られた通信端末数 4〜10 台における輻輳ウィンドウ値と ping により測定した end-to-end の RTT(図 6) を見てみると，輻輳ウィンドウは所々落ちているものの比較的安定して高い値を保っており，性能劣化に大きな影響を及ぼしているとは考えにくい．それに対して，RTT の遷移を見てみると人工遅延装置で設定した有線部の遅延時間の値よりもはるかに大きな遅延が観察された．

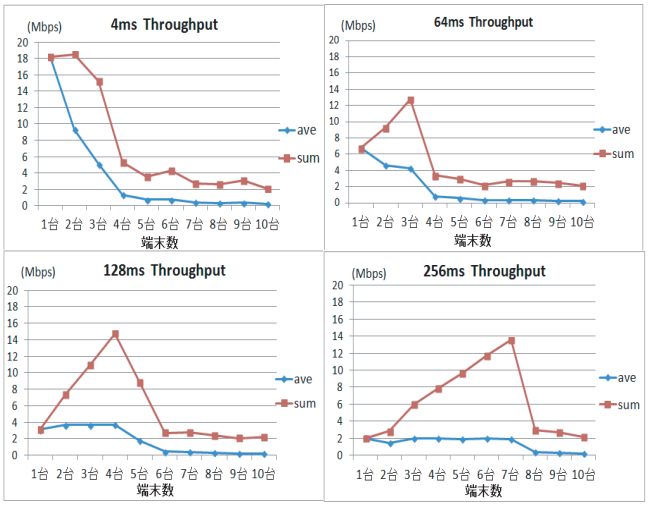

図 5　通信台数の変化によるスループットの平均値，合計値

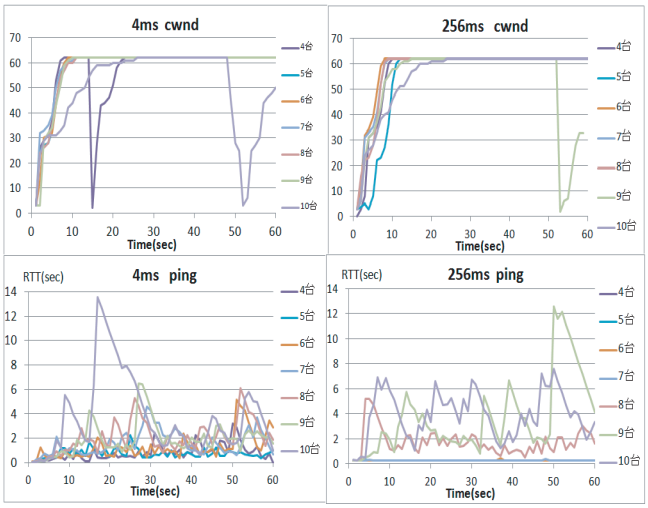

図 6　輻輳ウィンドウと実際の遅延時間 (RTT) の遷移

このことから本実験の考察として，RTT の大幅な増加が通信速度の低下につながるのではないかと予想できる．そこで，カーネルモニタで取得するパラメータに RTT とその最小値を追加した．ここで言う RTT とは，ネットワークの負荷を含むもの，つまり実際の遅延時間を指し，それに対し RTT の最小値はネットワークの負荷を含まない遅延時間，つまり dummynet で設定した人工的な遅延時間を表す．

これらを常時観察することで，現在のトラフィックの混み具合を把握できる．

5.　往復遅延時間と通信性能に関する検証実験

5.1　実 験 概 要

RTT が通信性能に与える影響を明らかにするために，前節と同じ実験環境において改変後のカーネルモニタを導入した Android 端末を用いた実験を行った．

5.2　実験結果と考察

有線部の人工遅延時間 16ms，128ms で Android 端末を 10 台で通信させた時のスループット，カーネルモニタで取得した RTT の遷移を図 7 に示す．グラフから，スループットが高いところでは遅延時間は小さく，逆にスループットが低いところでは遅延時間は大きいという対比が見られた．

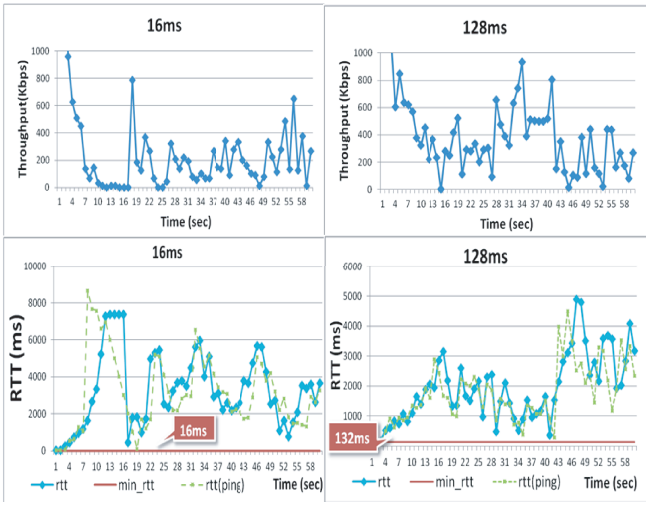

図 7　スループットと往復遅延時間 (RTT) の遷移

このことから，前節で予想した通り RTT の大幅な増加が通信性能劣化の大きな要因だと考えられる．よって，通信速度を向上させるためには，RTT の増減を考慮した制御が有効であると言える．

6.　提案ミドルウェア

6.1　ミドルウェアの構成

ここで，本研究で提案する通信制御ミドルウェアの概要を説明する．

改変前のミドルウェアでは，図 8 のように発信部と受信部に分かれて，それぞれで制御を行っていた．

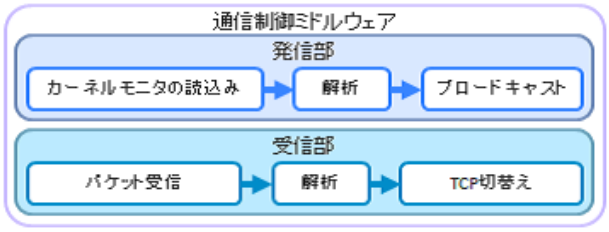

図 8　改変前のミドルウェアの構成

これに対し本研究では，実際に最適化チューニングを行う受信部においてもカーネルモニタの読み込みが必要となったため，この受信部と発信部を一括にまとめることで、導入や制御の簡単化を実現した．

図 9 に示すのは改変後のミドルウェアの構成である．通信中はカーネルモニタを常時監視し，RTT とその最小値を取得する．取得した値をもとに RTT/最小値で RTT の増減の比率 (ratio-rtt) を求める．また同時に，パケットを受信し，他端末の通信状況を把握してトラフィックを予測する．RTT の比率と通信台数の情報をもとに，外部プロセスから制御可能な proc インタフェースを用いて輻輳ウィンドウの上限値と下限値を設定し，最適化チューニングを行う．

これによって，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やそれに伴って急に RTT の値が増加したことをミドルウェアが検知すると，およそ 1 秒毎に輻輳ウィンドウ値を適切な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ことができる．

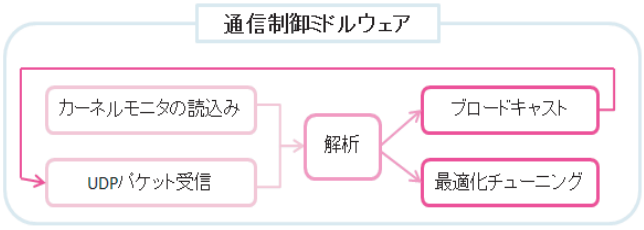

図 9　改変後のミドルウェアの構成

6.2　制 御 方 法

各遅延環境において，以下の式を用いることにより輻輳ウィンドウの上限値，下限値を設定した [8].

- 帯域幅遅延積 = 帯域幅 [Mbps] * 往復遅延時間 [sec]
- 輻輳ウィンドウの理想値 = 帯域幅遅延積 / 1 セグメントあたりのデータ量 [1.5Kbyte] / 通信端末数
- スループットの理論値 = 輻輳ウィンドウ値 * 1.5[Kbyte] * 8 / 往復遅延時間 [sec]

これらの式より算出した値をもとに設定した，通信端末数における制御と RTT の増減における制御のパラメータ値をそれぞれ表 2，表 3 に示す．また，RTT の最小値 (min-rtt) で以下の 4 段階に場合分けをしている．

A　：0 ≦ min-rtt < 10
B　：10 ≦ min-rtt < 100
C　：100 ≦ min-rtt < 200
D　：min-rtt ≧ 200

表 2　端末数における制御

端末数	A		B		C		D	
	max	min	max	min	max	min	max	min
1	21	20	63	62	63	62	63	62
2	16	15	63	62	63	62	63	62
3	11	10	46	45	63	62	63	62
4	9	8	36	35	63	62	63	62
5	7	6	28	27	55	54	63	62
6	5	4	23	22	46	45	63	62
7	4	3	21	20	41	40	63	62
8	3	2	17	16	36	35	63	62
9	2	1	15	14	31	30	61	60
10	2	1	14	13	28	27	56	55

さらに，端末数における制御と RTT の増減における制御をそれぞれ比較して，小さい方の値を採用している．

7.　提案ミドルウェアの評価実験

7.1　実 験 概 要

4 節と同じ実験環境において，本研究で開発したミドルウェアを Android 端末 10 台に導入し，その評価実験を行った．

表 3　RTT における制御

A			B			C			D		
ratio-rtt	max	min	ratio-rtt	max	min	ratio-rtt	max	min	ratio-rtt	max	min
1.0〜	63	62	1.0〜	63	62	1.0〜	63	62	1.0〜	63	62
10.0〜	61	60	2.0〜	62	60	2.0〜	58	57	2.0〜	56	55
15.0〜	58	55	3.0〜	56	53	3.0〜	55	54	3.0〜	51	50
20.0〜	53	50	4.0〜	48	45	4.0〜	46	45	4.0〜	41	40
25.0〜	48	45	5.0〜	42	40	5.0〜	36	35	5.0〜	31	30
30.0〜	41	40	6.0〜	36	34	6.0〜	26	25	6.0〜	21	20
35.0〜	36	35	7.0〜	26	25	7.0〜	16	15	7.0〜	11	10
40.0〜	31	30	8.0〜	21	20	8.0〜	11	10	8.0〜	6	5
45.0〜	26	25	9.0〜	21	20	9.0〜	6	5	9.0〜	4	3
50.0〜	21	20	10.0〜	11	10	10.0〜	4	3	10.0〜	2	1

7.2　実験結果と考察

合計性能を表すトータルスループットの結果を図 10 に示す．青がミドルウェアなしの状態で，赤が提案手法を用いた制御によるものである．人工遅延 64ms においては端末数 3 台，人工遅延 256ms においては端末数 7 台で大部分の帯域を活用できるようになる．グラフより，人工遅延 64ms のときにスループットが線形増加になったのは，輻輳ウィンドウの最小値を設定したことで高い値を保持することができたからだと考えられる．

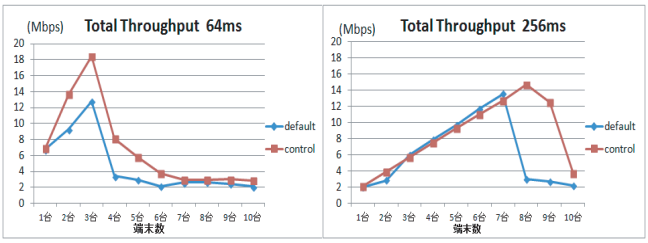

図 10　トータルスループット

さらに，人工遅延 64ms における 4〜6 台と人工遅延 256ms における 8〜10 台においては，RTT が小さいときは設定した輻輳ウィンドウの最小値を下回らないこと，また，RTT が大きいときには，設定した輻輳ウィンドウの最大値を上回らないことでその後の大幅な遅延を回避することができた．これらの適切な制御によって，通信速度が向上したものと考えられる．その振舞いを人工遅延 256ms の端末数 9 台におけるスループット，輻輳ウィンドウ，ratio-rtt の遷移として図 11 に示す．左軸がスループット，右軸が輻輳ウィンドウと ratio-rtt である．

以上の結果より，高遅延環境において特に本提案手法は効果的であると言える．

8.　まとめと今後の課題

本研究では，同時通信端末数が多い環境において合計通信速度が大幅に低下する問題に着目し，同一アクセスポイントにつながる周辺端末数に加える制御パラメータとして RTT を用いる手法についての考察を行った．基礎実験の結果から，同時に通信する端末数が多いときに全体の通信性能が劣化する直接的な原因は RTT の大幅な増加であることがわかった．そこで，その RTT の増加の比率を制御のパラメータとして取り入れ，よ

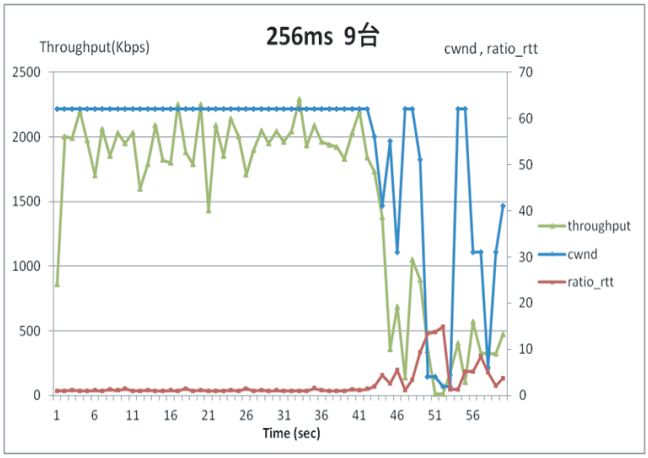

図 11　スループット，輻輳ウィンドウ，ratio-rtt の遷移

り最適な帯域見積りを行う通信制御ミドルウェアを開発した．その評価実験から，本提案ミドルウェアにより高遅延環境における通信端末数が多いときの全体の通信速度を向上させることに成功した．

今後の課題としては，さらなる高性能化を目指しより精密な最適化チューニングを行う．さらに，幅広い環境で性能向上のための制御手法を考案したい．また，研究目的である連携した通信制御を目指し，今後は各端末の RTT の増減情報を他端末と共有し，その情報に基づく協調的な制御を行う．さらに，ミドルウェアを持たない端末とが混在する環境において，本提案手法が通信性能にどのような影響を及ぼすのかを調査し，それに適応するための制御手法を考案する．

謝　　辞

本研究を進めるにあたり，ご指導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．

文　　献

[1] 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

[2] 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

[3] 平井弘実，山口実靖，小口正人: 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，DICOMO2012，2012 年 7 月．

[4] gartner:http://www.gartner.com/it/page.jsp?id=2237315

[5] android:developers:http://developer.android.com

[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] Iperf:http://downloads.sourceforge.net/project/iperf/iperf/2.0.4

[8] 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