モバイルネットワークにおける周辺端末からの情報に基づく協調制御ミドルウェアの提案と実装

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: †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性能化に伴い，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化した新しい通信制御の仕組みを提案する．TCP 輻輳制御は，エンドツーエンドのコネクションに対し，経路上の輻輳崩壊を回避するために 1 つの確認応答に対して送出するセグメント数を調節する役割を担っている．しかし，スマートフォンにおけるサービスは，主にクラウドサーバと連携することで高速処理やデータのバックアップなどを提供している．クラウドサーバは，WAN などの高帯域な有線経路によって支えられているため，スマートフォンからのアクセスにおいては，輻輳制御のボトルネックは低帯域な無線通信区間，すなわちエンドホストからアクセスポイントまでの区間に存在すると考えられる．本稿では，我々はスマートフォンにおいて協調的通信制御を行うミドルウェアを提案する．既存の輻輳制御は，有線経路を含む全体の経路上で起きた転送エラーに基づいてのみ輻輳崩壊を避けるためにフローを制御しているが，モバイル端末のインターネットアクセスにおいて性能劣化の主な原因は，無線区間に潜んでいるため，端末間が独立したエンド・エンドの輻輳制御は最適とは言い難い．そこで，本ミドルウェアは，アクセスポイント (AP) を共有するノード間で TCP のコネクション状況に関する情報を交換し，交換された情報を元にネットワーク負荷を判断し，TCP の処理を補正する．輻輳制御を補正することにより，各端末に公平な通信帯域を割り当てることが可能であるため，端末間の通信速度の公平性が向上する．さらに，この仕組みにより無線 LAN 内のトラフィックを効果的に制御することができるため，トータルスループットの向上も期待できる．我々は本稿において，このミドルウェアを Android 端末上に実装し，輻輳制御の協調的補正の有効性を示すため実機における評価を行った．

キーワード　Android，モバイルネットワーク，TCP/IP，輻輳制御

Proposal and Implementation on Cooperative Transmission Control Middleware on a Smartphone in a WLAN Environment

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: †hiromi@ogl.is.ocha.ac.jp, ††sane@cc.kogakuin.ac.jp, †††oguchi@computer.org

Key words　Android，Mobile network，TCP/IP，Congestion control

1. 研 究 背 景

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普及している．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いう位置付けであるため，汎用 PC のように動画の送受信やファイルの同期といった大規模なデータ通信を行う点が特徴である．しかし一方で，スマートフォンは，軽量化，及び小型化のために必要最小限のハードウェアリソースしか持たないため，処理能力や記憶容量が不十

分である．このデメリットを補うために，スマートフォン向けサービスは，クラウドサーバと連携して，データ処理や定期バックアップを行うといった仕様が標準になりつつある．

このような社会的背景から，スマートフォンの高速大容量通信に対する需要は高まっている．近年スマートフォンユーザが増えたことにより 3G 回線が圧迫されているが，帯域不足を克復するために，無線 LAN アクセスポイント (AP) の設置やモバイルルータの携帯が盛んである．そこで本研究は，スマートフォンの無線 LAN 利用時における通信性能の向上を試みる．

無線 LAN 利用時の通信性能劣化の原因として輻輳が挙げられる．TCP 通信には，輻輳崩壊を避けるために送り出すパケットの量を回線の状況に応じて調節する制御が存在するが，この輻輳制御は，PC のような汎用コンピュータを想定しており，これまで有線志向で設計・開発されてきた．しかし，スマートフォンのように無線通信を常とする端末においては，最適であるとは言い難い．何故なら，スマートフォンにおいては，近くの AP までは無線アクセスを行うものの，その先は，WAN 等の高帯域ネットワークに支えられたクラウドサーバに繋がっていることが多いためである [1] [2]．既存の輻輳制御は，エンドツーエンドの経路の状態だけを頼りに，送り出すセグメント数を決定しているが，図 1 に示すようなスマートフォンのネットワーク・トポロジにおいては，輻輳の可能性があるのは，スマートフォンと AP の間の低帯域なアクセス区間であると考えられる．本研究は，このようなスマートフォンのネットワークトポロジに最適化した輻輳制御手法の実現を目的とする．

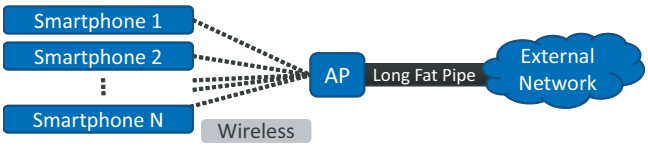

図 1　スマートフォンのネットワークトポロジ

1.1 関 連 研 究

輻輳崩壊を回避する制御として，TCP Cubic [3] が現在の Linux カーネルには採用されている．輻輳制御アルゴリズムを改良するために，近年では以下に述べる仕組みが議論されている．Large Initial Window [4] [5] は，通信コネクションの初期状態で，帯域幅を有効に使えていないことに注目し，輻輳ウィンドウの初期値を大きくすることが有効であると示したが，一方で複数端末が帯域を取り合っている場合については，考慮されていない．また，TCP Vegas [6] [7] は，損失ベースのアルゴリズムとは大きく異なり，理想の往復遅延時間 (RTT) と実際の RTT の差を元にネットワークの輻輳状態を見積もるアルゴリズムである．しかしながら TCP Vegas は，競合する全ての端末が，同じ遅延ベースアルゴリズムを採用している時にのみ効率的に働くが，一方で実世界においては大部分の端末は損失ベースのアルゴリズムを採用しているため，実用性が低いと考えられる．このデメリットを補うために Hybrid TCP [8] [9] や Compound TCP [10] は，損失モードと遅延モードを備えているが，周辺環境の情報が不十分であることから，常に効率的に動作する帯域幅遅延積の見積り手法は未だ明らかではない．

他の手段として，Explicit Congestion Notification [11] は経路上の不特定の地点においてパケットにビットを立てることで輻輳状態を通知する．また Active Queue Management [12] [13] は，輻輳崩壊が起きる前に，ルーティング機器上で意図的にパケットを損失させる仕組みである．これらの仕組みは差し迫った輻輳に備えて，セグメント送出量を減らすタイミングを検出する．しかし，これらの仕組みは，経路上の全てのネットワーク機器が対応する時のみ効果的であることから，通信インフラを新規に導入する必要があるためコストが大きく，実際には導入が進んでいない．

我々の提案は、輻輳の早期発見技術の一つであり，モバイル端末の変更のみで導入が可能であるため，容易に導入することができる．我々はこれまでに，いくつかの環境に対してチューニングされたアルゴリズムを切り替えるミドルウェアを開発した [14]．しかし，状況に合わせた多種のアルゴリズムを用意し，蓄積されたパターン情報に基づいてアルゴリズムを切替えるという手法は，現実世界への導入は必ずしも容易ではない．そこで本稿においては，我々は周辺環境に合わせて，既存のアルゴリズムを補正するミドルウェアを提案する．

1.2 Android OS

スマートフォン向けオペレーティングシステム (OS) 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携帯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 を採用する携帯端末メーカが増え，現在 Android はトップシェアとなった．Android の特徴として，従来の日本独自仕様の携帯電話と異なり，アプリケーションの開発と導入が容易であること，Linux Kernel を基盤とする，オープンソースプラットフォームであるためバックグラウンドの処理においても自由な開発が可能であることが挙げられる．

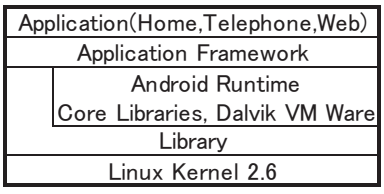

図 2　Android のアーキテクチャ

図 2 に示すように，Android は Linux Kernel を基盤とし，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 [15]．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．また Dalvik 仮想マシンとは別に，BSD 標準 C ライブラリの派生物である Bionic libc が搭載されているため，ネイティブバイナリも実行可能である．本研究が注目する通信制御は，カーネル内部で処理されているが，トランスポート層の制御においては，Android 独自の最適化はなく，Linux foundation [16] により配布される Linux Kernel のソースコードが利用されている．

2. 複数端末が AP を共有する時の挙動

本章においては，実際に複数端末で AP を共有し，帯域が混み合っている場合の輻輳制御の挙動を確認する．本稿にて示す実測値は，以下の図 3 に示す実験装置，及び表 1 に示す実験環境を用いたものである．通信性能を測定するベンチマークとして，Iperf for Android [17] を利用する．またクラウドサーバは，遠隔地のスマートフォンからアクセスされることが多いため，高遅延環境を想定し，往復遅延時間を 256ms とした．この遅延は，人工遅延装置 Dummynet [18] によって加えられる．

表 1　実 験 環 境

Client host	
Model number	GT-I9023
Firmware version	Android 4.1 (Jelly Bean)
Baseband version	I9023XXKD1
Kernel version	Linux 3.0.31
Build number	JRO03L
Server host	
Operating System	Ubuntu10.04.3
Linux version	2.6.32-37-generic-pae
CPU	Intel(R) Core(TM) i3 CPU 530@2.93GHz
Main Memory	3GB
Dummynet	
Operating System	FreeBSD 6.4-RELEASE
CPU	Intel(R) Pentium(R)4 CPU 3.20GHz
Access point	
Model number	MZK-MF300N
Manufacturer	PLANEX COMMUNICATIONS INC.
Legacy mode	IEEE802.11g

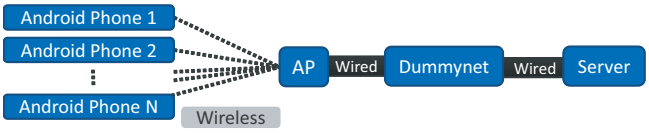

図 3　実 験 装 置

2.1 複数端末との AP 競合環境における通信性能の劣化

図 4 は，各端末数において AP を共有し同時に通信を行った時のトータルスループットを示す．端末数が最も低い 2 台で AP を共有している時は，UDP で約 25Mbps，TCP で約 19Mbps のトータルスループットを得ることができているが，いずれも端末数が増えると性能が劣化することが明らかである．また，端末数が少ない環境においては，全ての端末が UDP 通信を行っている時の方が，トータルスループットが高いが，端末数が多い環境では，TCP 通信を行った時のトータルスループットの方が高くなっており，この結果から端末数が多くなると輻輳制御は UDP に比べて帯域を効率良く活用できていると言える．TCP 通信はエラー率が高くなると輻輳と判断して，送出量を少なくするが，UDP はそのような判断を行わない．一般論として，TCP はパケット損失時に再送を行うため，通信速度が UDP に劣るが，端末数が多い環境においては，その逆の結果が得られた．この結果は，輻輳制御を効果的に行うと AP やネットワーク経路が有する全体のトラフィックが制御されるため，再送機能を持たない UDP よりも高速な通信性能を実現できることを示す．すなわち，輻輳制御をより効率良く行うことで，全体のトラフィックの量を調節できれば，トータルスループットは向上すると考えられる．輻輳制御が適切にセグメント数を調節し，可用帯域を使い切り，パケット損失が全く起こらない最も理想的な振舞いをしたと仮定した場合，端末数が増えても常に端末数が少ない時と同じ約 19Mbps の通信速度を維持することが可能と考えられ，実際の制御をこれに近づける．

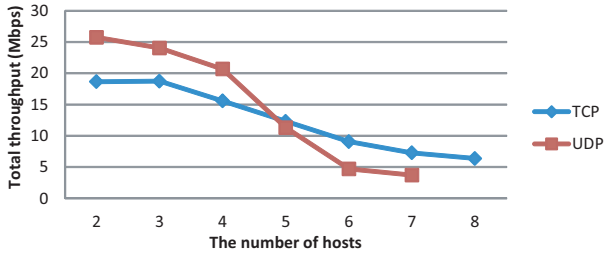

図 4　複数端末で AP を共有した際の通信速度

2.2 Android4.1 に搭載される標準の輻輳制御の振舞い

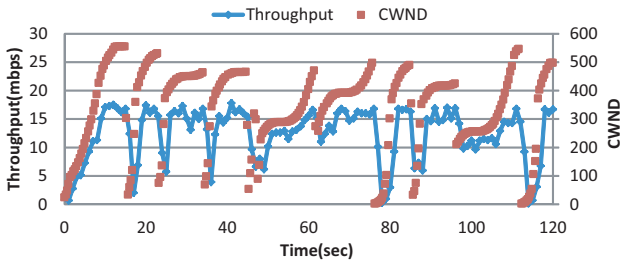

図 5　標準 TCP を利用する 1 台の端末で AP を占有した時の通信制御と通信速度の遷移

図 5 は，Android4.1 を起動した端末 1 台で AP を占有した時の輻輳ウィンドウの値とその時の通信速度を示す．この図から輻輳ウィンドウと通信速度には，相関関係あることがわかる．標準 TCP は，通信開始から少しづつ輻輳ウィンドウの値を増加させ，帯域の限界を探査する [19]．また，帯域の限界をパケット損失によって検出するとウィンドウの値を急激に減少させ，再度増加を試みている．この時，同時に通信速度も急減してい

ることから，コネクション確立時やパケット損失時からのウィンドウ増加中に未使用帯域を残してしまっていることがわかる．仮に TCP が予め AP を占有していることを認識していたなら，パケット損失を検出しても輻輳ウィンドウを急減させる必要はなく，その時点の未使用帯域を有効活用できたと考えられる [20].

2.3 輻輳制御の補正

輻輳ウィンドウの値は通信速度と相関性があるため，輻輳ウィンドウの値を操作することで，通信速度が間接的に操作できる．しかし，現状のシステム環境においても有線経路で輻輳が起こらない保証はないため，フレキシブルにウィンドウスケールする必要がある．そこで我々は，無線の可用帯域を上回るパケット送出を防止し，同時に経路全体の輻輳を回避するためには，エンドツーエンドで輻輳制御を行うデフォルトの輻輳制御アルゴリズムは活かしながら，これに加えて輻輳ウィンドウの最大値を設けることが望ましいと判断した．図 6 は，プロセスインタフェースに輻輳ウィンドウの上限値を表す変数を実装し，上限値を各値に設定した Android 端末が単独で AP を独占し，60 秒間の TCP 通信を行った際の通信速度を示す．本稿では，適切な帯域活用を行うために輻輳ウィンドウの上限値を設定する本手法を「輻輳制御の補正」と呼ぶ.

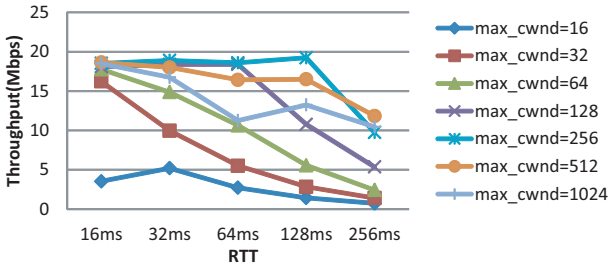

図 6　輻輳ウィンドウの上限値を各値に固定した端末の AP 占有時における通信速度

2.2 節より，TCP Cubic の輻輳制御は，エンドツーエンドのパケット損失によってのみ可用帯域を検出しているため，輻輳ウィンドウの大きさが定まらず，未使用帯域や帯域混雑を引き起こしていることが示された．スマートフォンのネットワークトポロジにおいて，無線 AP を共有する端末間で全体のトラフィックを調整することにより，輻輳によるパケット損失と再送及び，損失後の通信速度の低下を抑制することが可能だと考えられる．周辺環境が明らかであることを仮定すると，輻輳ウィンドウの理想値を次の式で計算することができる．N は端末数を示す.

$$Ideal\ cwnd = \frac{BDP - \alpha}{Segment\ size \times N}$$

IEEE802.11g の場合，理想的な最高転送速度は 54Mbps とされているが，現実世界で実際に観測可能な最高転送速度は 20Mbps 程度 [21] であるため，この値に遅延時間を掛けたものを帯域幅遅延積 (BDP) とする．また，α =0Kbits の時，この式は理想的な輻輳ウィンドウを求められるが，現実世界においては適度に帯域に余裕を持たせる必要がある．通信速度の実測値から，α =120Kbits が適切と判断し，この値を用いた．この式に基づき計算された輻輳ウィンドウの最適値を表 2 に示す.

表 2　輻輳ウィンドウの最大値を設定するパラメータ

N（端末数）	5	6	7	8
MAX CWND	69	57	49	43

表 2 に示される値を，輻輳ウィンドウの上限値とし，AP を共有する複数端末を同時に通信させ通信速度を測定した結果を図 7 に示す．さらに，Fairness Index を計算した結果を図 8 に示す．Farness Index [22] とは，計算方法を以下に示すが，公平性を示す指標であり，1 に近いほど高い公平性を示す.

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

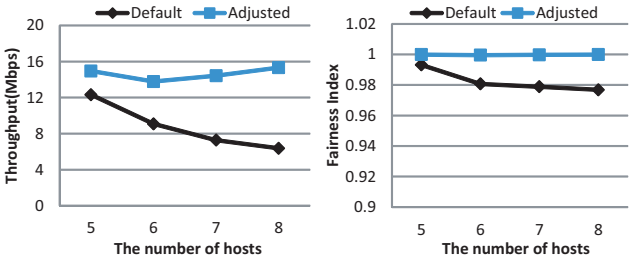

図 7　通信速度の比較

図 8　公平性の比較

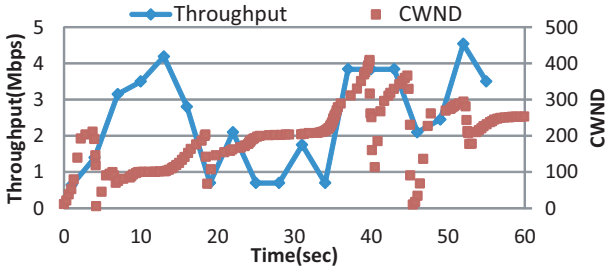

図 9　標準の輻輳制御を利用する 5 台の端末で AP を共有した時の通信制御と通信速度の遷移

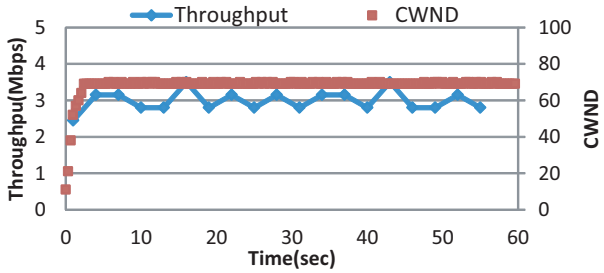

図 10　補正した輻輳制御を持つ 5 台の端末で AP を共有した時の通信制御と通信速度の遷移

さらに我々は，通信速度と公平性の向上が，輻輳制御の補正によって実現されたことを証明するために，端末数 5 台が競合通信を行ったうちの 1 台の端末の輻輳ウィンドウと通信速度の

遷移を解析した結果を図 9，10 に示す．補正した輻輳制御を利用した TCP 通信は，安定した適度な通信速度を維持していることがわかるが，一方で標準の輻輳制御を利用した TCP 通信は，通信速度は過度な増加と減少を繰り返している．また補正した輻輳制御に比べると，大きな輻輳ウィンドウを保持する場合が多いが，実際には輻輳ウィンドウが大きい時も通信速度は下がっていることがある．これは，複数端末が競合しトラフィックが溢れているために，MAC 層における CSMA/CA レベルでの衝突が多数発生し，再送が繰り返されたことが原因だと考えられる．

これらの測定結果 (図 7，8) と解析結果 (図 9，10) から，複数端末が競合する環境においては，上限値を設定することで輻輳ウィンドウを過度に増加させないことが，通信速度の向上に効果的であることが明らかである．また本手法により，各端末に利用可能な帯域を均等に割り当てられ，一部の端末が帯域を使いすぎることがなく，帯域をシェアすることができたため，公平性が向上したと考えられる．

3. 協調制御ミドルウェアの実装

2.3 節に示された輻輳制御の補正を効果的に行うには，周辺環境を認識し，適応する必要がある．本章では，AP を共有する端末同士が互いに情報交換を行い，収集した情報に基づいて，輻輳制御を補正する仕組みを示す．

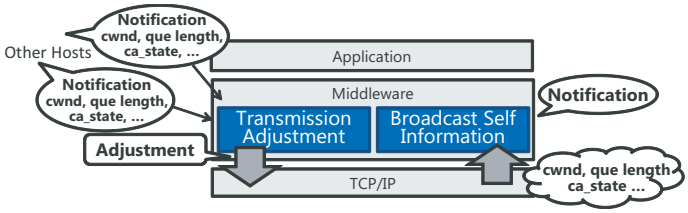

図 11　協調制御ミドルウェアの概念図

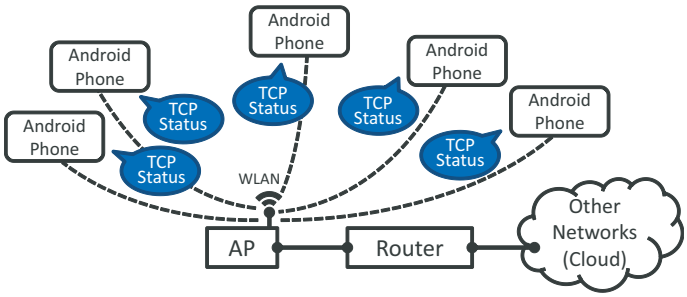

図 12　協調制御ミドルウェアの概念図

本ミドルウェアは，図 11 に示す 2 つのモジュールによって構成される．一方は，自らの通信処理を監視し，AP を共有する周辺端末に通信状況に関する情報を通知する役割を持つ情報共有モジュールである．このモジュールによる情報通知パケットは，UDP のブロードキャストを利用するため，AP を共有する全ての端末に向けて転送されるが輻輳時には破棄される．本稿では，情報通知の間隔は 0.5 秒とするが，この間隔は情報通知自体がオーバヘッドとならないよう柔軟な変更が可能である．この様子を図 12 に示す．また通知する情報の取得方法については 3.1 節で詳説する．

一方で輻輳制御補正モジュールは，周辺端末からブロードキャストされた情報を収集し，無線 LAN が保有しているトラフィック量を計算する．この計算結果から周辺状況を判断し，自らの輻輳制御を補正して，可用帯域を過度に活用することがないようにする．

本ミドルウェアは，C 言語で記述されたプログラムをクロスコンパイル [23] することで，Android 端末に導入している．ネイティブバイナリは，Android 独自の DalvikVM を利用しないため，バックグラウンドで端末に高負荷を与えることなく高速処理を行うことが可能である．

3.1 カーネル解析システムツール

協調制御ミドルウェアの一部である情報共有モジュールは，通信処理があった時にカーネル内部の通信状況に関する情報を解析し，周辺端末に提供する．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，ユーザ空間からその処理を監視することはできない．そこで本研究では，カーネル内部の通信処理を解析するために，カーネルモニタを Android に組込んだ．

カーネルモニタとは，Linux システムのカーネル内部の処理を解析する汎用 PC 向けシステムツールである．既存研究 [24] が，既にカーネルモニタの Android への組込みに成功しており，同様の方法で導入を行った端末を本研究においても利用する．

図 13 に示すようにカーネルモニタは、カーネルプロセスの監視及びパラメータのログの取得が可能である．カーネルモニタは様々な変数の値を取得可能とするが，本研究では TCP の処理にモニタ関数を組込み，輻輳ウィンドウ，RTT，ソケットバッファキュー長，CA 状態を取得可能とした．CA 状態とは，経路の状態を示すパラメータであり，正常の場合は Open，エラーを検出した場合は，それぞれ Disorder，CWR，Loss となる．取得されたパラメータはカーネル内のメモリ空間に記録され，プロセスインタフェースを介してミドルウェアが定期的にアクセスすることが可能となる．このような方法により，ミドルウェアは間接的に通信の手続きを監視する．

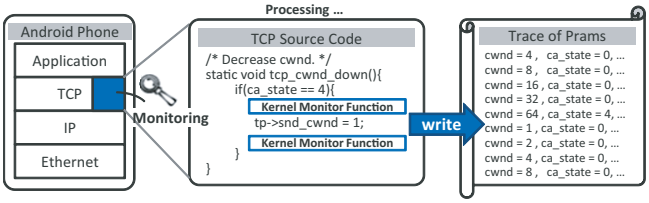

図 13　カーネル解析システムツール

4. ミドルウェアの評価実験

4.1 実験概要

本稿が提案するミドルウェアの効果を示すため，以下の評価実験を行った．本ミドルウェアの特徴は，周辺端末に通信状況に関する情報を通知する点と，周辺端末からの通知を受けて，

全体のトラフィックを予測し，輻輳制御を補正する点であるため，ミドルウェアが定期的にブロードキャストする通知がオーバヘッドにならないこと，アクティブに通信を行っている周辺端末数を通知パケットによって正確に認識できていることを示す必要がある．そこで，我々は，測定時間中にアクティブな送信端末数が 5 台から 8 台まで増える場合と，アクティブな送信端末数が 8 台から 5 台まで減る場合を想定し，実験 1 と実験 2 を行った．実験装置は，図 3 と同様とする．実験 1 と実験 2 のアクティブな端末数の時系列変化を図 14，15 に示す．実験時間は合計 240 秒であるが，60 秒間隔で端末が 1 台ずつ通信を開始もしくは終了し，アクティブな送信端末数は増加もしくは減少する．このような環境においてミドルウェアは，送信端末数が増える場合には，通知パケットを受信した時に通知元の IP アドレスを記憶し，送信端末数を更新する．一方で，送信端末が減る場合には，一定時間内に通知パケットを受けなかった IP アドレスを削除することで，記憶している送信端末数を定期的に更新している．本実験では，輻輳制御補正モジュールの端末数更新間隔は 10 秒とする．このような送信端末数が変化する環境におけるオーバヘッドを含めた通信速度を測定し，ミドルウェア導入による効果を示す．

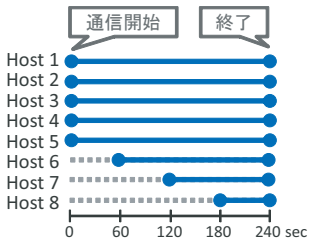

図 14　実験 1:端末数が増える場合

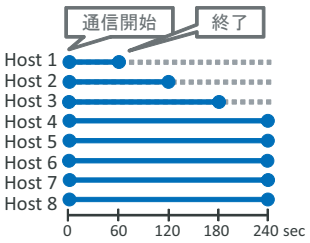

図 15　実験 2:端末数が減る場合

4.2 実験結果

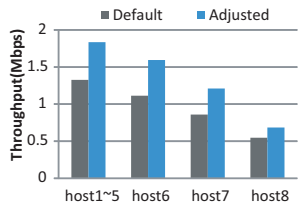

図 16　実験 1:端末数が増える場合

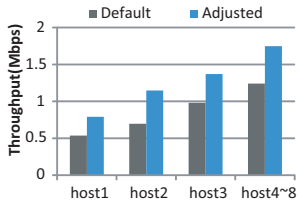

図 17　実験 2:端末数が減る場合

本評価実験の結果を図 16，17 に示す．アクティブな送信端末数が変化する環境下で，ミドルウェアを導入し，オーバヘッドを含めた通信速度を測定したところ，ミドルウェアを利用せず標準 TCP のみで制御した場合よりも，ミドルウェアを導入し輻輳制御を補正した場合の方が，全ての端末において，通信速度が向上している．

また，実験 1，2 の輻輳制御の振舞いを解析した結果を図 18，19，20，21 にまとめる．図 19，21 より，ミドルウェアを導入した場合，全ての端末数の輻輳ウィンドウの上限値は統一されており，各送信端末数に応じて表 2 に基づき正しく補正されたことがわかる．

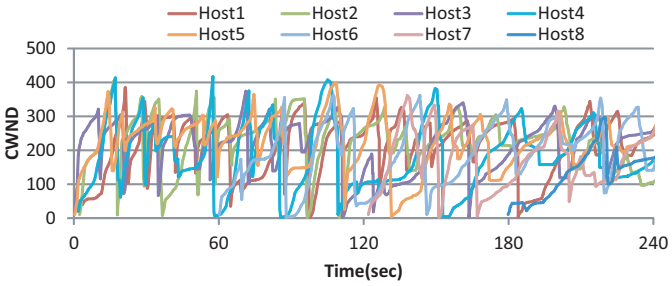

図 18　実験 1：標準 TCP を利用した端末の輻輳制御の振舞い

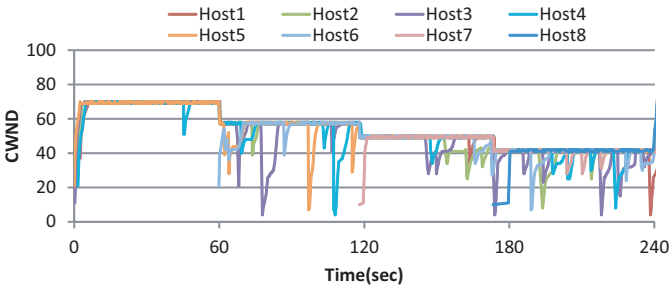

図 19　実験 1：ミドルウェアを導入した端末の輻輳制御の振舞い

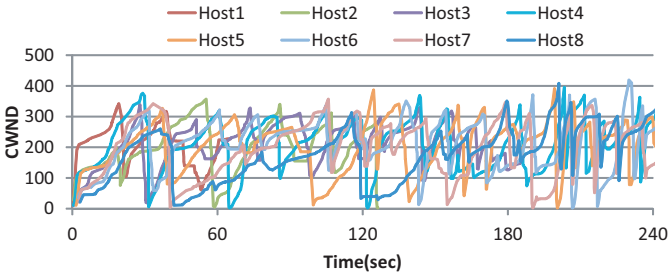

図 20　実験 2：標準 TCP を利用した端末の輻輳制御の振舞い

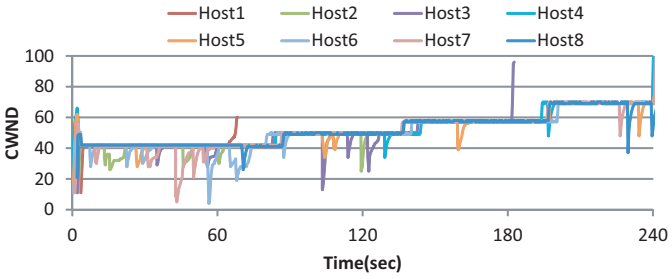

図 21　実験 2：ミドルウェアを導入した端末の輻輳制御の振舞い

さらに，これらの解析結果から，図 18，20 より，標準の輻輳制御を利用した時の輻輳ウィンドウの振舞いは，周辺端末との協調が取られず，緩やかな増加と急激な減少を繰り返していることが確認できる．一方で，ミドルウェアを導入した端末の輻輳ウィンドウの振舞いを観察すると，輻輳ウィンドウの上限値が制限されたことにより，安定したウィンドウサイズを保っており，転送エラーがほとんど発生していないことが明らかである．また，新たな周辺端末の通信開始時や通信終了時にも，ミドルウェアがすぐに適応し，効率的な通信が実現されている．よって，本提案ミドルウェアが複数端末の競合通信時に有効で

あることが示された．

5. まとめと今後の課題

標準的な輻輳制御アルゴリズムは有線ネットワーク向けに開発されてきたため、無線 LAN 環境下では常に良いとは限らない．標準の TCP コネクションは，端末間で互いに独立した制御を行うが，無線通信においてはノイズや電波干渉による任意のパケット損失があるため，損失ベースの Cubic は影響を受けやすく性能劣化が顕著である．また無線接続区間は有線接続区間に比べて低帯域であるため，ほぼ常に無線区間がボトルネックになっていると考えられる．

我々は，スマートホン市場でシェア率の最も高い Android 端末向けのミドルウェアを開発し，実際にその効果を示した．本ミドルウェアはカーネルモニタを利用して通信状況に関する情報を取得し，その情報をブロードキャストにより AP を共有する全ての端末に送信する．この情報通知により，環境に適応するため，ネットワーク上のトラフィック推測で輻輳制御を補正する．評価結果によれば，提案手法を導入した Android 端末上で通信速度と公平性が向上が確認できた．本稿では，Android プラットフォームにおいて，この協調的通信制御手法の有効性を証明したが，ミドルウェアを組込むことができれば，本手法は如何なるスマートフォンプラットフォームにおいてもその効果を発揮すると考えられる．

しかし，本ミドルウェアは，なお多くの課題を持つ．本稿では典型的なケースが紹介したが，自然環境には様々な状況が存在する．

（1） スマートフォン端末は海外などの遠隔地に存在するクラウドサーバにアクセスすることが多いが，常にその前提が成り立つとは限らない．そのような場合は，各 TCP コネクションごとに個別の RTT に基づいて，帯域幅遅延積を計算し，通信帯域を埋めつつも公平性を維持できる制御が必要となる．

（2） AP を共有する全ての端末が常に奪い合っているわけではない．本稿の実験においては，ベンチマークソフトウェアを利用し，大きなデータを転送する端末を実験装置としたが，現実世界においては，メールの送信等の帯域をさほど必要としない低トラフィックな通信を行う端末も存在する．このような端末にも公平に帯域を割り当てることは，かえって未使用帯域を残すことに繋がる．そこで，ソケットバッファキュー長を確認し，ユーザの通信需要に合わせた帯域割り当てを行うことが効果的であると考えられる．

（3） 本稿では，アップリンクについて言及し，実装ならびに評価を行ったが，アップリンクと同様ダウンリンクにおいても輻輳による通信性能劣化の問題は存在する．ダウンリンクについても MAC 層に頼るのではなく，トランスポート層においてパケット溢れを防止する対策が有効であると考えられる．

（4） 本稿では，AP を共有する全ての端末がミドルウェアを持つ時の評価を行ったが，ミドルウェアを持たない端末との混在環境については言及していない．そのような環境においても実験を行い，輻輳制御の補正により，かえって通信性能が低下してしまうことがないことを確認する必要がある．万一，そうなってしまった場合には，ミドルウェアを持たない端末数が一定割合を超えた時に，輻輳制御の自動補正を停止させ，標準の輻輳制御を利用すべきである．そうすれば，本ミドルウェアは常に現状維持もしくは，標準よりも良い通信性能を提供することが可能である．

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．

文　　献

[1] Feng, Wu-chun, Pavan Balaji, Chris Baron, Laxmi N. Bhuyan, and Dhabaleswar K. Panda. ”Performance characterization of a 10-Gigabit Ethernet TOE,” Proceedings of IEEE Symposium on High Performance Interconnects - HOTI , pp. 58-63, 2005.

[2] Chase, Jeffrey S., Andrew J. Gallatin, and Kenneth G. Yocum. ”End system optimizations for high-speed TCP,” Communications Magazine, IEEE 39, no. 4 (2001): 68-74.

[3] Sangtae Ha, Injong Rhee and Lisong Xu, ”CUBIC: A New TCP-Friendly High-Speed TCP Variant,” Proceedings of SIGOPS Operating Systems Review, 42(5):64-74, 2008.

[4] Dukkipati, Nandita, Tiziana Refice, Yuchung Cheng, Jerry Chu, Tom Herbert, Amit Agarwal, Arvind Jain, and Natalia Sutin. ”An argument for increasing TCP ’s initial congestion window,” ACM SIGCOMM Computer Communication Review 40, no. 3 (2010): 27-33.

[5] Allman, Mark, Chris Hayes, and Shawn Ostermann. ”An evaluation of TCP with larger initial windows,” ACM SIGCOMM Computer Communication Review 28, no. 3 (1998): 41-52.

[6] Brakmo, Lawrence S., Sean W. O’malley, and Larry L. Peterson. ”TCP Vegas: New techniques for congestion detection and avoidance,” Vol. 24, no. 4. ACM, 1994.

[7] La, Richard J., Jean Walrand, and Venkatachalam Anantharam. ”Issues in TCP vegas,” Electronics Research Laboratory, College of Engineering, University of California, 1999.

[8] Kaneko, Kazumi, Tomoki Fujikawa, Zhou Su, and Jiro Katto. ”TCP-Fusion: a hybrid congestion control algorithm for high-speed networks,” Proceedings of Protocols for Future, Large-Scale & Diverse Network Transports (PFLDNeT), 2007.

[9] Masafumi Hashimoto, G. Hasegawa, M. Murata. ”Performance Evaluation and Improvement of Hybrid TCP Congestion Control Mechanisms in Wireless LAN Environment,” Proceedings of Telecommunication Networks and Applications Conference, Australasian - ATNAC , 2008.

[10] Song, Kun Tan Jingmin, Q. Zhang, and M. Sridharan. ”Compound TCP: A scalable and TCP-friendly congestion control for high-speed networks,” Proceedings of Protocols for Future, Large-Scale & Diverse Network Transports (PFLDNeT), 2006.

[11] Floyd, Sally. ”TCP and explicit congestion notification,” ACM SIGCOMM Computer Communication Review 24, no. 5 : 8-23, 1994.

[12] Firoiu, Victor, and Marty Borden. ”A study of active queue management for congestion control,” Proceedings of IEEE International Conference on Computer Communications (INFOCOM), vol. 3, pp. 1435-1444, 2000.

[13] Floyd, Sally, and Van Jacobson. ”Random early detection gateways for congestion avoidance,” IEEE/ACM Transactions on Networking - TON , vol. 1, no. 4, pp. 397-413, 1993.

[14] Hiromi Hirai, Kaori Miki, Saneyasu Yamaguchi, and Masato Oguchi, ”A Study on Transmission-Control Middleware on an Android Terminal in a WLAN Environment,” Proceedings of the Fourth International Conference on Wireless, Mobile Networks & Applications (WiMoA-2012), pp.469-479, Delhi, India, May 2012.

[15] Android open source project, `http://source.android.com`

[16] The Linux Kernel Archives, http://www.kernel.org/
[17] Iperf For Android Project in Distributed Systems, http://www.cs.technion.ac.il/~sakogan/DSL/2011/projects/iperf/index.html
[18] The dummynet project, http://info.iet.unipi.it/~luigi/dummynet
[19] Stevens, W.R.: "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," The Internet Society (RFC2001), 1997.
[20] Xylomenos, George, George C. Polyzos, Petri Mahonen, and Mika Saaranen. "TCP performance issues over wireless links," Communications Magazine, IEEE 39, no. 4 (2001): 52-58.
[21] De Bruyne, Jeffrey, Wout Joseph, Leen Verloock, and Luc Martens. "Evaluation of link performance of an indoor 802.11 g network." Proceedings of Consumer Communications and Networking Conference (CCNC), 5th IEEE, pp. 425-429. IEEE, 2008.
[22] 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.
[23] Sourcery G++ Lite for ARM GNU / Linux, http://www.codesourcery.com
[24] Miki, Kaori, Saneyasu Yamaguchi, and Masato Oguchi. "Kernel Monitor of Transport Layer Developed for Android Working on Mobile Phone Terminals," Proceedings of The Tenth International Conference on Networks (ICN), pp. 297-302. 2011.