情報処理学会論文誌 Vol.55 No.1 340–353 (Jan. 2014)

スマートフォンの無線LAN接続時における周辺端末からの情報に基づく協調帯域制御ミドルウェアの提案と実装

平井 弘実[1,†1,a)] 山口 実靖[2,b)] 小口 正人[1,c)]

受付日 2013年4月10日, 採録日 2013年10月9日

概要：近年スマートフォンの爆発的な普及と高性能化にともない，スマートフォンによる高速データ通信に対する期待が高くなっている．本研究は，このような期待に対応するため，スマートフォンの無線 LAN 利用時のデータアップロードに最適化させた新しい通信制御の仕組みを提案する．現行の TCP の輻輳制御アルゴリズムは，有線通信指向で開発されており，各端末が独立して，送出するセグメント数を見積もっていたため，ノイズの多い無線通信においては，最適な制御を行っているとはいい難く，課題が多く残されていた．そこで本稿では，アクセスポイント（AP）を共有している周辺端末を連携させ，互いの情報を利用して，より精密な可用帯域幅の見積りを行うミドルウェアを開発し，実機の Android 端末に導入して通信性能が向上することを示した．

キーワード：スマートフォン，モバイル端末，無線 LAN，TCP/IP，輻輳制御，通信帯域制御

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

Hiromi Hirai[1,†1,a)] Saneyasu Yamaguchi[2,b)] Masato Oguchi[1,c)]

Received: April 10, 2013, Accepted: October 9, 2013

Abstract: In recent years, the digital convergence era has arrived. Computers are required to be conveniently usable anywhere and anytime. The great demand for portable computers has produced the smartphone, which replaces PDA or the cellular phone. The traditional TCP congestion control plays a critical role of flow control in the end-to-end connection. In the case of access from a wireless device such as smartphone, however, the bottleneck of data flow almost always lies in the wireless connection, i.e. packet transmission between mobile hosts and access point. This is because, for nowadays, a wired connection such as Gigabit Ethernet has a considerably wide bandwidth compared to a wireless one. In this paper, we present middleware for cooperative transmission control on smartphones. The middleware allows each mobile host connected to a wireless LAN AP to exchange information of its own TCP connection status and performs a cooperative control to increase the communication throughput as well as to improve the fairness for bandwidth utilization. We implemented the middleware on the Android platform and evaluated it with physical devices to show the effectiveness of the cooperative adjustment for the congestion control in this paper.

Keywords: smartphone, mobile device, wireless LAN, TCP/IP, congestion control, transmission bandwidth control

[1] お茶の水女子大学
Ochanomizu University, Bunkyo, Tokyo 112–8610, Japan
[2] 工学院大学
Kogakuin Uniersity, Shinjuku, Tokyo, 163–8677, Japan
[†1] 現在，日本電信電話株式会社
Presently with NTT
[a)] hiromi@ogl.is.ocha.ac.jp
[b)] sane@cc.kogakuin.ac.jp
[c)] oguchi@computer.org

1. はじめに

近年，スマートフォンが世界中で急速に普及している．スマートフォンは，小型コンピュータであり，その多くが片手に収まる大きさであるため，携帯性に優れ多くのユーザに親しまれている．スマートフォンは，高速データ通信に対応したデバイスであるため，動画によるコミュニケー

ションや人の生活を支援するセンサとしての先端的な利用方法も注目されている．このようなリアルタイム性が重要となるサービスは，常時十分な帯域確保が必要となるため，ネットワークの混雑を上手に回避する必要がある．

一方で，近年スマートフォンユーザが増え，大量のトラフィックが飛び交うようになったため，通信キャリアは3G 回線の帯域不足を克服するために，全国各地で人が集まる主要地点に無線 LAN アクセスポイント（AP）の設置を積極的に進めている．しかし，1 台の AP によって吸収できるトラフィックには限界があり，さらにその限界を高めようと 1 カ所に複数の AP を設置した場合には，かえってチャネル間干渉による通信不良を招いてしまう [1]．

これらの不都合を緩和する手段の 1 つとして，1 台の AP を効率良く利用するために送信端末側の工夫は有効である．TCP/IP 通信を利用するコンピュータシステムは，オペレーティングシステムを問わず，共通のプロトコルスタックを保持し，第 4 レイヤであるトランスポート層は輻輳制御という，ネットワーク上の混雑を回避する送信端末側の帯域制御機構を有する．輻輳制御とは，パケットが通過するそれぞれの経路の状態にかかわらず，送受信の両端のノード間の状態に応じて，通信量を調整する仕組みである．輻輳制御機構には，受信側からの確認応答（ACK）が返って来るまでの間に送出するセグメント数を示す輻輳ウィンドウというパラメータがあり，輻輳制御アルゴリズムによって適切な輻輳ウィンドウの大きさを推測しているが，いかなる状態においても最適なウィンドウスケールを行うアルゴリズムはいまだ見つかっていない．特に無線通信においては，自然環境のノイズとネットワークの輻輳を正確に見分けることは非常に困難であり，ノイズによるパケット損失により送信量を自粛してしまった場合は，可用帯域を十分使いきることができない．

現在のスマートフォンの利用環境を考えると，端末にインストールされたアプリケーションを利用し，サービスを提供するサイトにアクセスするケースが非常に多く，これは主にクラウドサーバに対する遠隔アクセスとなっている．このような場合，**図 1** に示すように，AP から先は有線区間であり，高帯域のネットワークにつながっているため [2], [3]，モバイル端末とサーバの間でボトルネックとなるのは，大部分が無線区間であると考えられる．そこで本研究は，ボトルネックになりうる無線区間において，AP を共有する端末間で各々の通信状況に関する情報を交換し，端末間が連携した協調的制御を行うミドルウェアを提案する．

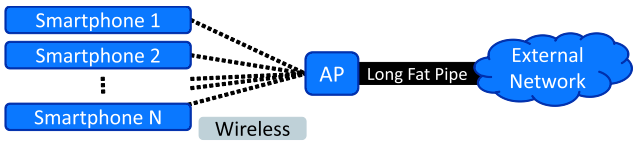

図 1 スマートフォンのネットワークトポロジ
Fig. 1 Smartphone network topology.

現在の Linux カーネルの輻輳制御アルゴリズムには，損失ベース方式の TCP Cubic [4] が標準で採用されている．TCP Cubic 以前の損失ベース方式アルゴリズムは，正常な ACK を受け取ると，輻輳ウィンドウを少しずつ増加させ，非正常 ACK により通信エラーを検出すると，輻輳ウィンドウを大きく減少させることが一般的であった．この手法では低遅延環境にアクセスするフローの輻輳ウィンドウは迅速に増加するが，高遅延環境にアクセスするフローの輻輳ウィンドウは緩やかに増加するため，フロー間に不公平が生じるという問題があった．この点を補うために，TCP Cubic は通信エラー検出からの経過時間によって輻輳ウィンドウを調節するという仕組みを採ったため，時間の経過とともに輻輳ウィンドウが回復するという利点を持つ．しかし，1 度ネットワークが輻輳した後，回復を待たずに輻輳ウィンドウを増加させてしまうという欠点を持ち，最適な輻輳ウィンドウを保つことの難しさと，これを end-to-end のみで制御する手法の限界が示されているといえる．

しかしながら，この問題を解決するために，既存の方式とはまったく異なる新たなトランスポート層の制御アルゴリズムを開発し，実世界の端末に導入することは，親和性の点で現実的とはいい難い．輻輳制御アルゴリズムの多様化は，つねに無数のアルゴリズムとの相性を考慮することになってしまう．また，AP を改造し，輻輳制御のために特別な機能を持たせたものに置き換えていくことは，設備投資のコストが大きいため広く受け入れられることが期待できない．したがって，他の端末の通信性能にほとんど影響を与えず，モバイル端末側のみの改変で対応可能な手法が望ましい．さらにモバイル端末に関しても，カーネルコードの TCP 実装そのものを改変してしまうと互換性が低下するため，カーネルコードは既存のものとし，これをミドルウェアで補正し対応する手法が有効であると考えられる．

本研究は，TCP Cubic のアルゴリズムが可用帯域を上回るトラフィックを送出することを防ぐために，輻輳ウィンドウに上限値を設定しアルゴリズムを補正するという仕組みである．輻輳ウィンドウは，その大きさによって送出されるデータ量が帯域幅遅延積（Bandwidth Delay Product: BDP）の 2 倍以上であることが，ネットワークパイプを無駄なく埋めるために理想的といわれているが，実測値によれば，TCP Cubic は帯域幅遅延積の数十倍まで輻輳ウィンドウを増加させてしまうことがあり，混雑を引き起こす原因となり，通信性能は劣化することが分かった．

すなわち，TCP Cubic は過度にウィンドウを増加させ，AP でパケット溢れが発生し性能劣化を引き起しやすい傾向があるため，この積極性を適切に抑えるミドルウェアが望ましい．本研究の提案するミドルウェアは，コネクショ

ンを張った帯域幅と片道遅延時間を掛け合わせ，その値をAP を共有する周辺端末数で割った値を，端末 1 台あたりの可用帯域幅遅延積ととらえ，その 2 倍の値を輻輳ウィンドウの最大値として設定することで，無線ネットワークの許容量を上回る輻輳ウィンドウの過度な増加を防止するという仕組みである．現状の通信制御方式に調整を加え，無線区間における過多な通信パケットを抑制する手法であることから，有線区間での輻輳が起こった場合には既存の制御アルゴリズムが適切に働く．さらには，同じ AP を共有する端末数とその端末によって生じるトラフィックの量は変動するため，周辺端末の状況に応じて，ネットワークの許容量のうち，ある端末が使える帯域幅を公平に算出したうえで，輻輳ウィンドウの最大値を変動させる必要がある．この仕組みにより，複数の端末が協調して効率良く 1 台の AP を共有することで，通信スループットの向上を示す評価実験を行った．本稿では，端末が一定時間継続的にデータ通信を行う場合を想定し，通信性能の向上を示したが，今後の課題として，細切れの低トラフィックなデータ通信を行うフローも共存する場合における適切な通信制御も検討の範囲と考えている．すなわち，本稿の提案手法を拡張することにより，様々なトラフィック環境においても効果的に通信制御を行うことができると考えられる．

このような制御手法をミドルウェアとしてスマートフォンのような小型端末に導入することは，常時バックグラウンドで働き続けるため，端末の CPU 負荷を気にする必要がある．場合によってはミドルウェアの CPU 負荷により，かえって性能が低下する可能性が懸念される．筆者らの実験においても，初期の Android 端末（HTC Magic）ではハードウェアリソースが十分ではなかったため，ミドルウェアを導入しても十分な性能向上が達成できないこともあったが，近年のスマートフォンの性能向上は目覚ましく，この傾向は今後も続くものと考えられる．したがって，現世代のスマートフォンのリソースが十分であるなら，今後も有望なアプローチであるといえよう．

本研究による貢献は以下の 3 点である．まず 1 番目は，現在広く用いられているスマートフォンの無線 LAN 環境における通信状態とその問題点を明らかにしたことである．無線通信は，端末の性能と通信機器の性能，用いられている通信プロトコルの仕様などが頻繁に変わり，それぞれ性能が大きく向上している．しかし全体の通信性能は，これら個々の要素がうまく組み合わさらなければ向上しないため，実際にはそれぞれを動作させ確認してみる必要がある．本稿ではまずこのような実験を行い，現状で用いられているスマートフォンの TCP 実装は，必ずしも無線環境で良い性能を出せるようにはなっていないことを明らかにし，その理由についての解明を行った．2 番目の貢献としては，スマートフォンのようなモバイル端末において，既存のプロトコル実装はそのまま用い，これを補正することにより適切に動作するためのミドルウェアを動かす方法について明らかにし，これを実際に実装して動作させたことである．新たなプロトコルを導入したり既存のプロトコルを改変したりするのではなく，標準的なプロトコルの動作をモニタリングして対処するミドルウェアを機能させたことにより，今後対象となるスマートフォンの実装がバージョンアップしていったとしても，本稿の提案手法はそのまま適用できる可能性が高い．3 番目の貢献としては，提案手法を実機の Android 端末に実装して動作させ，評価を行った結果，本提案手法の有効性を実環境で示したことである．詳細は後述するが，本ミドルウェアの導入により，全ノードのトータルスループットに加え，各端末ごとのスループットの公平性も向上させることに成功した．さらに，本ミドルウェア実装を持たない他の送信端末が同一 AP 上に混在する環境においても，本ミドルウェアの有効性を示すことができた．

本稿の構成は以下のとおりである．まず 2 章において，関連研究や現在のスマートフォンの OS などの研究背景について説明した．次に 3 章においては，複数のスマートフォンで 1 台の AP へ接続しデータ通信を行ったときの通信状況を，カーネル実装の内部パラメータである輻輳ウィンドウなどの振舞いまで含め詳しく解析し，問題点を指摘した．4 章においては，本研究で提案するミドルウェアの実装方法について，解説を行った．そして 5 章においてミドルウェアの評価実験を行い，提案ミドルウェアが複数端末の競合時に有効であることが示された．さらに 6 章では，提案ミドルウェアを実装していないシステムとの混在環境においても，本ミドルウェアの導入により望ましい通信環境となったことを説明している．最後に 7 章でまとめと今後の課題について述べている．

2. 研究背景

2.1 関連研究

本節では，輻輳回避を目的とした関連研究を紹介する．通信経路上のネットワーク機器による制御では，以下の研究があげられる．Explicit Congestion Notification [5], Explicit Loss Notification [6] は，経路上で軽度の輻輳を検出した時点でネットワーク機器がパケットにビットを立てることで状態を通知し，深刻な輻輳を回避する．Active Queue Management [7], [8] 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，輻輳ウィンドウを減少させるタイミングを検出する．また，モバイル端末に特化した仕組みとしては，I-TCP [9]，Snooping TCP [10] があげられる．通常，TCP は送信者と受信者の end-to-end のコネクション状態によって送出するパケットの量を調整しているが，無線区間と有線区間の通信特性は大きく異なるため，I-TCP はこのコネクションを 2 つに分

割し，無線通信に最適化した TCP プロトコルを無線 AP とモバイル端末に導入することで高速化を行う．Snooping TCP は，無線 AP だけを改造し，無線 AP 自身が ACK 発信やパケットの再送を行う仕組みである．しかし，これらの仕組みは，経路上のすべてのネットワーク機器が対応するとき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設備投資のコストが大きく，実際に導入が進んでいない．

そこで，送信側の改造のみで実装可能な送信帯域制御手法が期待されている．送信帯域制御手法の 1 つとして，クロスレイヤ制御による技術が存在する．文献 [11], [12] は，MAC 層の情報としてチャネル利用率を参照し，その値に応じて，輻輳ウィンドウの大きさを適切に設定し，利用帯域幅を調節するという仕組みである．輻輳ウィンドウの大きさによって送信帯域制御を行う点は，本研究の提案方式と一致するが，参照する情報が，その端末自身の共有する無線チャネル利用率である点が異なる．一般的に，プロトコルスタックの各層は独立することが望ましいため，クロスレイヤ制御はこのような指針に反する．トランスポート層の制御は，トランスポート層内の情報を基に制御できることが望ましい．

輻輳制御アルゴリズムを改良する手法として，近年では以下に述べる仕組みが議論されている．TCP Vegas [13], [14] は，損失ベース方式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 の差をもと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，競合するすべての端末が，同じ遅延ベースアルゴリズムを採用しているときにのみ効率的に動作するが，一方で実世界においては大部分の端末は損失ベース方式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．損失モードと遅延モードを備えて切替えを行うハイブリッド型 TCP [15], [16] や Compound TCP [17] においても，他のプロトコルの影響を受けやすいことが欠点としてあげられる．

輻輳ウィンドウは，BDP の 2 倍に設定することが理想的であるが，BDP は AP を共有する周辺端末台数や周辺端末の通信量の変化によって変動するため，ユーザの移動によって AP を共有する端末数が変わりやすいモバイルネットワークにおいては，十分な注意を払うことが通信性能の向上に結びつくと考えられる．文献 [18] は，受信側から返される ACK から現在の BDP を推定し，推定した BDP および BDP の変動に基づいて過度なセグメントが送信されないように輻輳制御を行う Flight Size Auto Tuning を提案している．

本研究では，遅延時間はコネクションに依存するものの，AP の帯域幅はつねに一定であると考え，AP を共有する周辺端末間で情報交換し，端末間で AP の提供する帯域幅を公平に分けるミドルウェアを提案している．このような周辺端末と連携した通信制御を実装し通信性能向上に成功した実験例は，著者らの知る限りでは世界的にも初めてである．

我々の提案手法は，輻輳の早期発見技術の 1 つであり，周辺端末の通信状況に関する情報を輻輳検出の手がかりとする．さらに送信端末の変更のみで導入が可能であるため，実世界への導入が比較的容易である．我々はすでに，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 [19]．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り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，我々は次に周辺環境に合わせて，自端末の利用可能な帯域幅を計算し，輻輳ウィンドウを一定値に固定するミドルウェアを開発した [20]．しかし，実験環境におけるスループットは大きな向上が見られたものの，ミドルウェアを持つ端末間の情報を基にして一定値に固定した輻輳ウィンドウは，ミドルウェアを持たない端末の帯域を奪ってしまうことがあった．本稿で提案する仕組みは，輻輳ウィンドウのスケールに最大値を加えるだけであるため，ネットワークの許容量を明らかに上回るトラフィックの送出を防止することで，オーバフローを回避しており，幅広い環境に適応可能かつ，異なる通信デバイスとの親和性を保つことが可能である．

また，Large Initial Window [21], [22] は，通信コネクション確立時に，スロースタートで輻輳ウィンドウを 1 から増加し始めると，可用帯域を大きく余らせていることに着目し，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．本稿では，通信中の輻輳ウィンドウの最大値をもって公平かつ効率的な帯域割当てを提案するが，通信確立時のウィンドウ初期値は大変興味深い．今後の課題として，周辺端末と連携した適切なウィンドウ初期値の設定についても検討したい．

2.2 モバイル 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，すでにいくつかの種類が存在するが，その中でも一際注目を集めたのは Android OS [23] である．Android OS は，Linux Kernel [24] を基盤とし，オープンソースプラットフォームであることを特徴とし，開発者にとっては幅広く自由な開発が可能であり，端末メーカにとっては，システム開発を大きく削減できることが大きな魅力であることから，近年急速に成長をとげて世界トップシェアとなった．Android が，スマートフォン市場において一般ユーザ向け Linux ディストリビューションを提供したことは大きな貢献である．Android の成長とともに，ARM アーキテクチャ向けに最適化した Linaro [25] の発展に加わったため，今後のスマートフォン市場に様々な OS が登場した

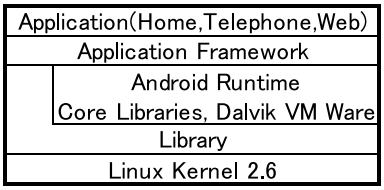

図 2 Android のアーキテクチャ
Fig. 2 Architecture of Android.

としても，Linux ベースのものが高いシェア率を維持することが期待できる．本稿において，我々は Android OS をブートした実機の携帯端末によるミドルウェア実装と評価実験例を紹介するが，Linux ベースの OS においては，PC を含めても導入が容易である．

図 2 に示すように，Android は独自の Runtime を持ち，その上で動作するアプリケーションが一般的であるが，Open BSD 由来の C ライブラリを持つため，ネイティブコードを実行することも可能である．ネイティブコードは Dalvik VM を利用しないため，メモリ消費が少なく高速な処理が可能であるため，本研究ではネイティブコードで実装したミドルウェアをクロスコンパイルし，実機の Android 端末に導入している．クロスコンパイラとして，Sourcery G++ Lite for ARM GNU [26] を利用した．

3. 複数端末が 1 台の AP を共有するときの挙動

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．本稿で示す実測値は，図 3 に示す実験装置，および表 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 [27] 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 256 ms とした．この遅延は，人工遅延装置 Dummynet [28] によって加えられる．

3.1 複数端末の競合時における通信性能の劣化

複数端末が 1 台の AP を共有する場合，端末数が増えるごとに，端末 1 台あたりのスループットが下がることは避けられない．しかし，端末からのアップリンク方向におけるデータ転送は特に，周辺端末の行動が予想できないため，AP を共有した端末が同時にデータ転送を行ったときのスループットを足し合わせた値であるトータルスループットにおいても劣化することが分かっている．

図 4 は，各端末数において AP を共有し同時にデータ通信を行ったとき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 を共有しているときは，UDP で約 25 Mbps，TCP で約 19 Mbps 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ごと

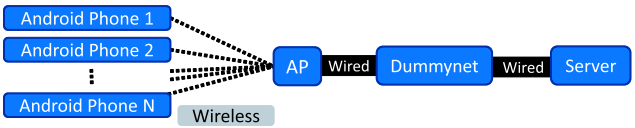

図 3 実験装置
Fig. 3 Experimental system.

表 1 実験環境
Table 1 Experimental environment.

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@2.93 GHz
Main Memory	3 GB
Dummynet	
Operating System	FreeBSD 6.4-RELEASE
CPU	Intel(R) Pentium(R)4 CPU 3.20 GHz
Access point	
Model number	MZK-MF300N
Manufacturer	PLANEX COMMUNICATIONS INC.
Legacy mode	IEEE 802.11g

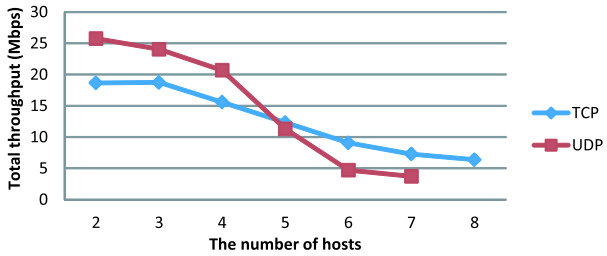

図 4 複数端末で AP を共有した時のトータルスループット
Fig. 4 Total throughput when AP is shared by multiple hosts.

にトータルスループットが減少し，通信効率が悪くなってい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，UDP 通信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 通信のトータルスループットの方が高くなっている．この結果から，端末数が多い場合は，輻輳制御機構を持つ TCP は，輻輳制御を行わない UDP と比べると帯域を効率良く活用できているといえる．一般論として，TCP はパケット損失時にパケットの再送を行うため，スループットが 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，輻輳制御機能を持たない UDP よりも高速な通信性能を実現できることを示す．

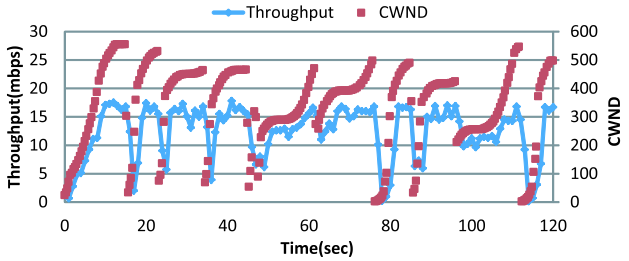

図 5 1 台の端末が AP を占有した時の輻輳ウィンドウとスループットの遷移

Fig. 5 Transition of congestion window and throughput when a single host exclusively accesses to AP.

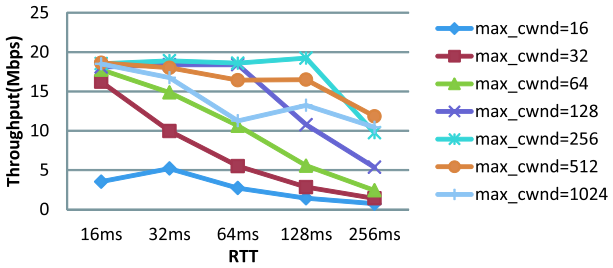

図 6 輻輳ウィンドウの上限値を各値に設定した端末の AP 占有時におけるスループット

Fig. 6 Throughput of each host in the case that maximum value is set to congestion window when a single host exclusively accesses to AP.

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，端末数の多いときのトータルスループットはさらに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，可用帯域を使いきり，パケット損失がまった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，端末数が増えてもつねに約 19 Mbps のトータルスループットを維持することが可能と考えられるため，現実的に可能な範囲で，そのような理想の状態に近づけるような通信制御手法を提案する．

3.2 デフォルトの TCP 利用時の輻輳制御の振舞い

図 5 は，端末 1 台で AP を独占したときの輻輳ウィンドウの値とそのときのスループットの実測値を示す．Android 端末のリソースが限られていることと，IEEE802.11g を利用していることから，有線でネットワークにつながった PC に比べると，輻輳ウィンドウは比較的小さな値を保っているが，このような場合においても，輻輳ウィンドウとスループットには，相関関係あることが分かる．TCP 輻輳制御は，通信開始から少しず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 [29]．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．このとき，同時にスループット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分かる．仮に TCP があらかじめ AP を独占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 [30]．

3.3 輻輳制御の補正

輻輳ウィンドウの値はスループットと相関性があるため，可用帯域に対して輻輳ウィンドウが小さい場合には，スループットも低くなる．すなわち，ネットワークが許容する限りは，輻輳ウィンドウの値を意図的に大きくすることで，スループットを向上させることができる．しかし，輻輳ウィンドウが大きすぎてネットワークの許容量を上回る場合には，オーバフローによりパケットが破棄されるので，かえってスループットを低下させてしまうことになる．さらには，パケット破棄が起きると，輻輳ウィンドウが急減するため，スループットが自主的に抑制されてしまうことになる，

そこで我々は，可用帯域を上回るパケット送出を防止するためには，周辺端末と協調して輻輳ウィンドウの制限値を設けることが望ましいと判断した．なぜなら，周辺端末の通信量を知ることで，事前にパケットのオーバフローを防ぐことができるからである．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 は，輻輳ウィンドウの上限値をそれぞれの値に設定した端末が，競合端末なしに AP を独占し，データ転送を行ったときのスループットを示す．本実験は RTT を 16 ms から 256 ms まで変化させて実験を行った．輻輳ウィンドウの上限値は，16 から 256 までの間では，大きくするほどスループットが向上するが，512 を超えるとかえってスループットが低下してしまっている．

TCP Cubic は，通信エラーの種類と通信エラーの発生後の経過時間によって，輻輳ウィンドウの値を決定しているため，輻輳ウィンドウが大きすぎてパケット溢れを引き起こしてしまうことがあるが，可用帯域をパケットで埋めきることが可能なウィンドウサイズを最大値とすることで，パケット溢れを防止することが可能だと考えられる．パケット溢れを起こしにくい輻輳ウィンドウの大きさは以下の式で求めることが可能である．なお，N は端末数を指す．

$$Ideal\ cwnd = \frac{BDP - \alpha}{Segment\ size \times N} \quad (1)$$

IEEE802.11g の場合，理想的な最高伝送速度は 54 Mbps とされているが，TCP データ通信における実効速度は 20 Mbps 程度 [31] 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．遅延時間は，ネットワークの混雑具合いによっても左右されるため，同じセッションのパケットの中で最も短いものを参照する．通常は，周辺端末に関する情報は得ることができないが，情報を得る手段を実装したら，この式で輻輳ウィンドウの最大値を求めることができる．また，$\alpha = 0$ Kbits のとき，この式は

理想的な輻輳ウィンドウの最大値を求められるが，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と考えた．そこで，スループットの実測値から，$\alpha = 120$ Kbits 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大値を**表 2** に示す．

α はノイズなどによるトラフィックの急激な一時的変動を吸収するための値として機能する．$\alpha = 0$ Kbits の場合，BDP を周辺端末で分け合ってぎりぎりまで使いきっている形になり，実環境では瞬間的にトラフィックが変動した場合など，これに影響されて不必要にパケット送出を絞ってしまう可能性がある．これを防ぐため，BDP から少し値を引いた値を用いると安定することが実験により分かった．どの程度の値を用いればよいか，様々な値を用いて検証を行ったところ，上記のように $\alpha = 120$ Kbits で多くの場合に安定した性能向上が見られた．またこの値は，帯域幅が 20 Mbps で遅延が 16 ms とした場合であっても BDP の 1%未満であるため，この α の導入自体が全体の性能低下に影響を与える割合はきわめて低いといえる．

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データ通信をさせトータルスループットを測定した結果を**図 7** に示す．さらに，通信性能の評価値としてスループットのフェアネスインデックスを計算した結果を**図 8** に示す．フェアネスインデックス [32] 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．

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

たとえ複数端末で同時にデータ通信をしたときのトータルスループットが高かったとしても，一部の端末のスループットだけが高く，残りの端末のスループットが低い場合には，通信性能の向上とはいい難い．トータルスループットとフェアネスインデックスを同時に評価することによって，より良い通信性能を各端末のユーザに提供できる．この実験結果によれば，輻輳制御を補正したことで，端末数 4〜10 台が同時にデータ通信を行った場合にはトータルスループットとフェアネスインデックスの両方が向上することを確認できた．一方で，端末数 2〜3 台の場合には，トータルスループットは向上しているものの，フェアネスインデックスは 1%未満の劣化が見られた．端末数 3 台の場合を分析すると，実験に使った 3 台の端末の中で，最もスループットの低かった端末どうしを比較しても，補正によってスループットは 0.95 Mbps 向上している．すなわち，最もフェアネスインデックスの劣化が大きかった端末数 3 台の場合においても，補正によってスループットが劣化した端末はなく，3 台すべてのスループットが向上したことを確認した．さらに我々は，人工遅延時間を 16 ms，32 ms，64 ms，128 ms とした場合にも，同様の結果が得られることを確認している．

さらに我々は，トータルスループットとフェアネスインデックスの向上した原因が，本当に輻輳制御の補正によるものであることを証明するために，端末数 5 台が同時にデータ通信を行ったときの 1 台の端末の輻輳ウィンドウとスループットの遷移を解析した．その解析結果を**図 9**，**図 10** に示す．補正した輻輳制御を利用した TCP 通信は，安定した適度なスループットを維持していることが分かるが，一方でデフォルトの輻輳制御を利用した TCP 通信では，スループットが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

表 2 設定された輻輳ウィンドウの最大値
Table 2 Maximum value of congestion window.

N（端末数）	2	3	4	5	6	7	8	9	10
Max CWND	173	115	86	69	57	49	43	38	34

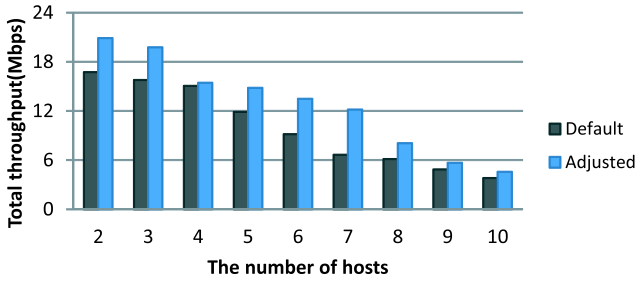

図 7 トータルスループットの比較
Fig. 7 Comparison of total throughput.

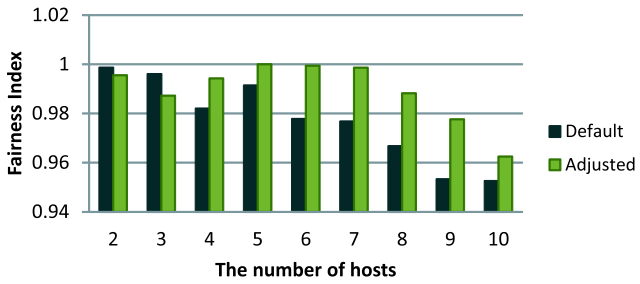

図 8 フェアネスインデックスの比較
Fig. 8 Comparison of fairness index.

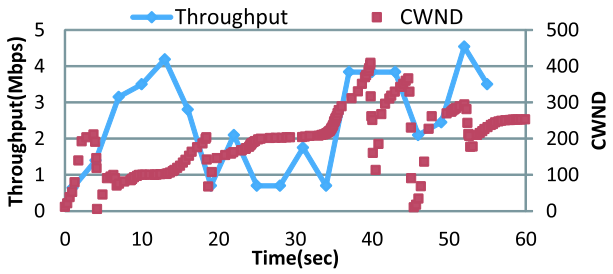

図 9 デフォルトの輻輳制御を利用する 5 台の端末で AP を共有した時の輻輳ウィンドウとスループットの遷移
Fig. 9 Transition of congestion window and throughput with default congestion control when AP is shared by 5 hosts.

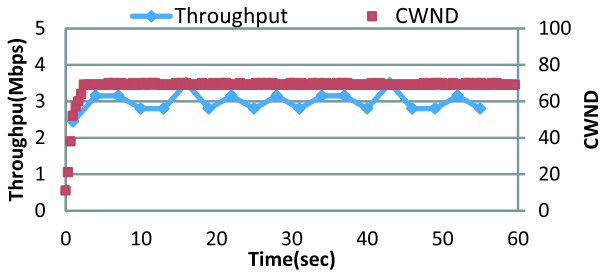

図 10 補正した輻輳制御を持つ 5 台の端末で AP を共有した時の輻輳ウィンドウとスループットの遷移

Fig. 10 Transition of congestion window and throughput with adjusted congestion control when AP is shared by 5 hosts.

いときもスループット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，周辺端末の情報を手に入れ，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，スループットの向上に効果的であることが明らかである．またフェアネスインデックスの向上については，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なったために，帯域を均等にシェアすることができたためだと考えられる．

4. 協調制御ミドルウェアの実装

3.3 節に示された輻輳制御の補正を効果的に行うには，周辺端末の存在を認識し，その環境に適応する必要がある．携帯端末は，ユーザの移動によって，無線 LAN に接続したり，切断したりするので，有線環境に比べて端末数が変わりやすい．本章では，AP を共有する端末どうし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．

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．

4.1 情報通知手段

本ミドルウェアの情報通知モジュールは，2 つの役割を持っている．1 つ目の役割は，自端末のカーネル内部の処理を常時監視し，送信処理があったときに送信状況に関する情報を取得することである．そして，2 つ目の役割は，得られた情報を効果的に周辺端末に通知することである．

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 に組み込ん

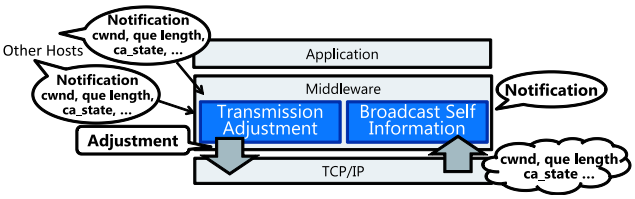

図 11 協調制御ミドルウェアの概念図

Fig. 11 An overview of cooperative control middleware.

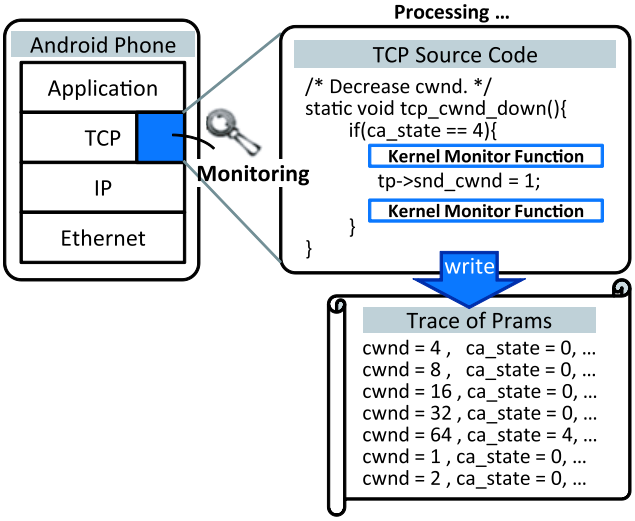

図 12 カーネル解析システムツール

Fig. 12 System tool for kernel analysis.

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 [33] が，すで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．

カーネルモニタは，**図 12** に示すように，カーネルプロセスの監視およ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み込み，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．これらのパラメータは，今後，より厳密に周辺端末と協調した通信制御を実現するうえで価値のあるデータとなる．輻輳ウィンドウの大きさにより，これから送出しようとしているパケットの量をある程度予測することはできるが，ウィンドウサイズの大小のみから一概にネットワークの混み具合いを推定することは難しいからである．輻輳ウィンドウが小さくても，ソケットバッファキュー長が短い場合には，そもそもその端末は大きなデータを送ろうとしていない可能性が高い．しかし，ソケットバッファキュー長が十分大きく，輻輳ウィンドウは小さく，CA 状態が Open でない場合には，ネットワークが混み合っている可能性が高い，といった判

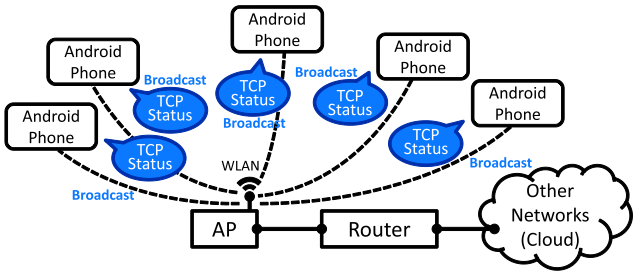

図 13 協調制御ミドルウェアの情報共有手段
Fig. 13 Information sharing using cooperative control middleware.

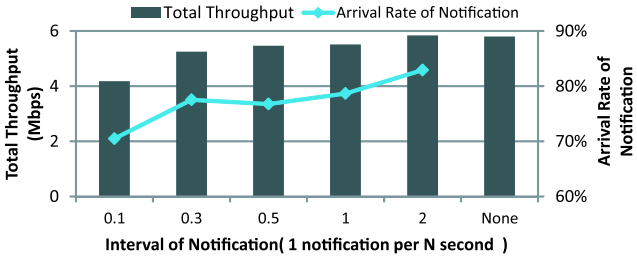

図 14 情報通知のオーバヘッドと到達率
Fig. 14 Arrival rate of notification at each interval.

断を今後活用することが考えられる．さらに，RTT が大きいフローと RTT の小さいフローが同時に流れている場合には，それぞれに必要な帯域幅は RTT に基づいて計算するといった帯域幅割当て方法も，今後実現できる可能性が考えられる．これらのパラメータはカーネル内のメモリ空間に記録され，カーネルモニタのプロセスインタフェースを介してユーザ空間からアクセスすることが可能となる．このような方法により，ミドルウェアは通信の手続きを監視し，情報を取得する．

カーネルモニタによって得られた情報は，**図 13** に示すように，UDP のブロードキャストで周辺端末に通知される．この通知は，ネットワークが混雑しているときに，周辺端末と協調するために必要であるが，この通知そのものがオーバヘッドになり性能が低下することは避けなければならない．ある端末の送信処理が続いている間は，通知パケットが定期的にブロードキャストされるが，この通知間隔が短すぎる場合は，オーバヘッドとなりうる．一方で，通知間隔が長すぎる場合には，十分な情報共有ができないというトレードオフの関係がある．そこで，情報通知パケットが，データ通信にどのような影響を与えるか検証した結果を**図 14** に示す．この検証結果は，各通知間隔を設定した情報通知モジュールを 8 台の端末に導入し，データ通信を行ったときのトータルスループットと，そのときの通知パケットの到達率を測定したものである．UDP パケットは，混雑時に破棄されてしまうため，到達率も重要な評価指標である．本実験において互いに情報通知は行っているが，それに基づく輻輳制御の補正は行っておらず，オーバヘッドが性能に与える影響のみを調べるための測定である．

一番右のグラフである情報通知モジュールの導入前に比べて，0.1 秒間隔で通知を行ったときは，トータルスループットが約 2 Mbps 弱程度低下しており，通知パケット到達率も 70%で高いとはいい難い．一方で 0.3 秒以上の間隔で通知を行ったときは，トータルスループットはあまり劣化しておらず，通知パケットの到達率も比較的高い．そこで，これらの中でも最も通知頻度が高く，データ通信への影響が少ない 0.3 秒が情報通知間隔として適切と判断した．通知間隔が 0.3 秒のときは，トータルスループットは約 1 Mbps 未満程度の劣化をするが，図 7 によれば，輻輳制御の補正により 2 Mbps 以上向上するため，結果としては，トータルスループットが 1 Mbps 以上向上したことになる．これは端末数が 8 台の場合のオーバヘッドであるため，端末数が 8 台以下のときには，オーバヘッドはより小さくなると考えられ，情報通知を用いることによりデータ通信の速度は，十分向上することが期待できる．また，端末数が 9 台以上で，通知がオーバヘッドとなってしまう場合においては，端末数によって通知間隔を変化させる仕組みを実装すれば，この問題は回避することができる．

なお情報通知パケットのブロードキャストのデータ量については，たとえば端末数が 8 台の場合，送受信データの全体のトラフィックに対して約 1/200 となっており，情報通知による送受信のオーバヘッドとなるデータ量は少量であるといえる．

4.2 輻輳制御補正手段

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通知パケットを収集し，AP を流れているトラフィックの量を計算する．この計算結果から周辺状況を判断し，3.3 節に示した計算方法により，自らの輻輳制御を補正して，パケット溢れが起こりにくいようにする．周辺状況とは，本稿ではアクティブにデータ送信を行っている周辺端末数を指すが，受信した通知パケットの IP アドレスを記憶することで，この情報を得ている．

通知パケットは 0.3 秒間隔で送られるが，UDP の到達率が 100%ではないため，通知間隔以上の一定間隔で更新される必要がある．端末数 8 台のときに，通知パケットの到達率は 78%であることから，すべての端末の情報を得られる可能性は 78%の 8 乗であるため，14%程度となる．すなわち，約 7 秒間隔で情報を更新することで，すべての端末の通知パケットを収集することができると考えられる．新規の IP アドレスから通知パケットを受信した場合には，その時点ですでにカウントしていた周辺端末数に 1 を足すが，周辺端末がデータ通信を終了したことに気づくことは容易ではない．そこで，一定時間内に届いた通知パケットを解析し，周辺端末数を更新することで，周辺端末がデータ通信を終了した，もしくは同じ無線 LAN エリアから脱出したことを認識する．この一定時間は，理論的には 7 秒が最

適と考えられるが，今回は，より正確性を重視して 10 秒とした．今後は，この端末数更新間隔をより短くし，環境の変化により敏感に対応できるようにしたいと考えている．

5. ミドルウェアの評価実験

5.1 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輻輳制御を補正する点であるため，アクティブにデータ送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，ミドルウェアによる通信性能の向上がオーバヘッドを上回っていることを示す必要がある．そこで，我々は，測定時間内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 15**，**図 16** 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うえでのスループットを測定し，ミドルウェア導入による効果を示す．

5.2 実験結果

本評価実験の結果を**図 17**，**図 18** 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スループットを測定したところ，ミドルウェアを利用せずデフォルトの 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，すべての端末において，スループットが向上している．

また，実験 1，2 の輻輳制御の振舞いを解析した結果を**図 19**，**図 20**，**図 21**，**図 22** にまとめる．図 20，22 より，ミドルウェアを導入した場合，すべての端末数の輻輳ウィンドウの上限値は，周辺端末数の変化するごとに補正されており，表 2 の数値に一致するため，周辺端末数は正しく認識されていたことが分かる．

さらに，これらの解析結果から，図 19，21 より，デフォルトの輻輳制御を利用したときの輻輳ウィンドウの振舞いは，周辺端末との協調がとられず，緩やかな増加と急激な

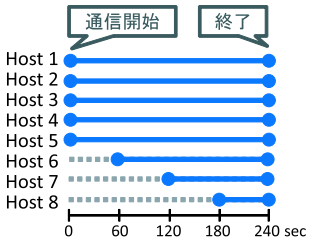

図 15 実験 1 における各端末の通信時間

Fig. 15 Communication period of each host in Experiment 1.

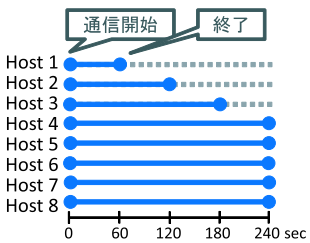

図 16 実験 2 における各端末の通信時間

Fig. 16 Communication period of each host in Experiment 2.

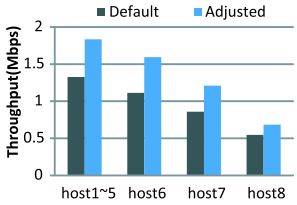

図 17 実験 1 のスループット

Fig. 17 Throughput in Experiment 1.

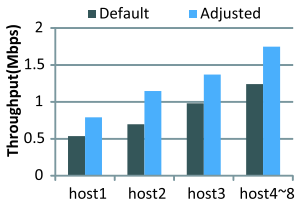

図 18 実験 2 のスループット

Fig. 18 Throughput in Experiment 2.

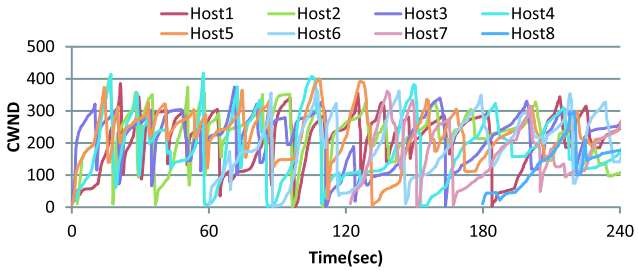

図 19 実験 1：デフォルトの TCP を利用した端末の輻輳制御の振舞い

Fig. 19 Experiment 1: Behavior of congestion window with default TCP.

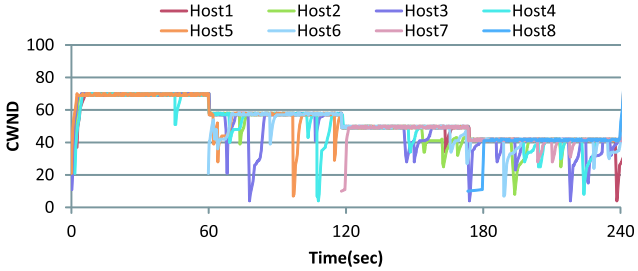

図 20 実験 1：ミドルウェアを導入した端末の輻輳制御の振舞い

Fig. 20 Experiment 1: Behavior of congestion window with middleware.

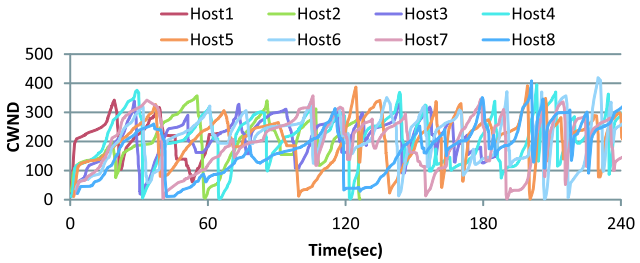

図 21 実験 2：デフォルトの TCP を利用した端末の輻輳制御の振舞い

Fig. 21 Experiment 2: Behavior of congestion window with default TCP.

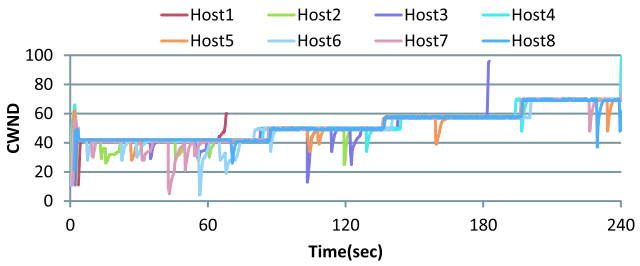

図 22 実験 2：ミドルウェアを導入した端末の輻輳制御の振舞い

Fig. 22 Experiment 2: Behavior of congestion window with middleware.

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，ミドルウェアが迅速に適応し，効率的なデータ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．

6. 従来環境との親和性

5 章では，AP を共有するすべての端末がミドルウェアを持つときの評価を行ったが，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．現実的に本ミドルウェアの導入を検討すると，他端末との親和性は無視できない．特性の異なる他の通信フローとの混在により，輻輳制御の補正がかえって通信性能が低下させてしまう，もしくは，既存手法を利用する端末の通信帯域を大きく奪ってしまうことも好ましくない．ミドルウェアは，つねにデフォルトよりも良い通信性能，もしくは現状維持できることが必要である．

6.1 ミドルウェアを持たない送信端末との混在環境におけるミドルウェア評価

輻輳制御を補正された端末と補正されていない端末が 1 台の AP を共有して，同時にアップリンクのデータ転送を行った場合の影響を明らかにすることが重要である．AP

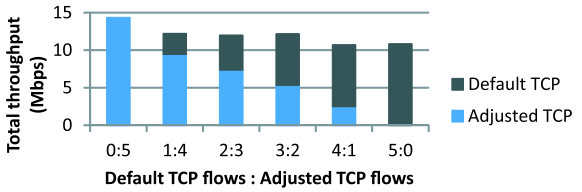

図 23 ミドルウェアを持つ端末とミドルウェアを持たない端末の各割合におけるトータルスループット

Fig. 23 Total throughput at each fraction of numbers of hosts with and without middleware.

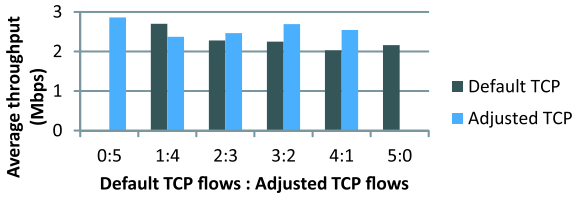

図 24 ミドルウェアを持つ端末とミドルウェアを持たない端末の各割合におけるスループットの比較

Fig. 24 Average throughput at each fraction of numbers of hosts with and without middleware.

を共有する合計端末数が 5 台のときに，補正された端末の数と補正されていない端末の数の割合を変化させ，スループットを測定した．**図 23** は，それぞれの環境のトータルスループット，および，補正された端末のトータルスループットと補正されていない端末のトータルスループットの内訳を示す．このグラフから，補正された端末の割合が高いほど，トータルスループットが向上していることが分かる．また，**図 24** は，それぞれの環境ごとの，補正された端末，および，補正されていない端末のスループットを示す．補正された端末が 1 台も存在しない場合における，補正されていない端末のスループットに比べて，ほぼすべての場合で，補正された端末にも補正されていない端末にもスループットの向上が見られる．このグラフから，一部の端末を補正したことによって，補正した端末はもちろん，補正していない端末にとっても利益があることが分かった．

6.2 UDP データ送信端末との混在環境におけるミドルウェア評価

次に，UDP でデータ転送を行う端末との混在環境における，TCP でデータ転送を行う端末へのミドルウェア導入の有効性を示す実験を行った．6.1 節と同様，AP を共有する合計端末数を 5 台とし，TCP 送信端末と UDP 送信端末の割合を変化させた．そのときのすべての端末のスループットを足し合わせたトータルスループットを**図 25** に示すが，ミドルウェアを導入し補正された TCP 送信端末が UDP 送信端末と競合したときの方が，おおむねミドルウェアを持たない TCP 送信端末が UDP 送信端末と競合したときよりもトータルスループットが高いことが分かる．

一方で，平均スループットについては，TCP と UDP は，

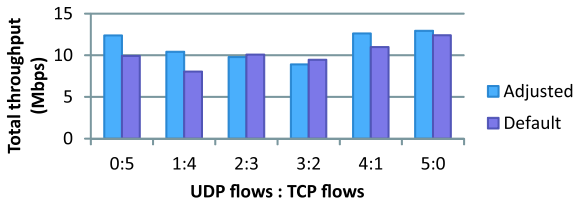

図 25 TCP フローが UDP フローと混在した時のトータルスループット

Fig. 25 Total throughput with mixture of TCP and UDP flows.

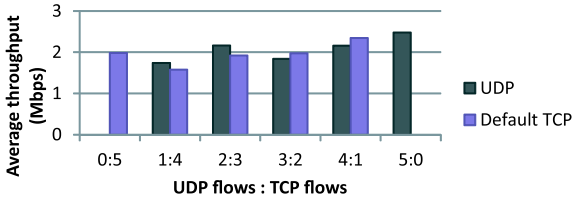

図 26 デフォルトの TCP フローと UDP フローの混在環境における各平均スループット

Fig. 26 Average throughput with mixture of default TCP and UDP flows.

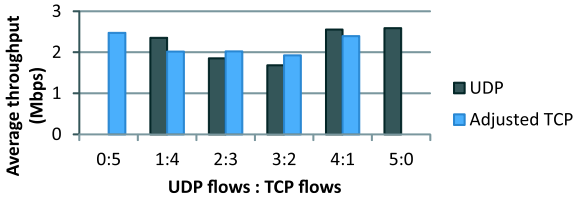

図 27 補正した TCP フローと UDP フローの混在環境における各平均スループット

Fig. 27 Average throughput with mixture of adjusted TCP and UDP flows.

それぞれ性質が大きく異なるため，それぞれのスループットを比較することはできない．そこで，ミドルウェアを持たない TCP 送信端末が UDP 送信端末と混在した場合の平均スループットと，ミドルウェアを持つ TCP 送信端末が UDP 送信端末と混在した場合の平均スループットを測定し，比較する．ミドルウェアを持たない TCP 送信端末と UDP 送信端末の混在環境における平均スループットを**図 26** に示し，ミドルウェアを導入した TCP 送信端末と UDP 送信端末の混在環境における平均スループットを**図 27** に示す．ミドルウェアを導入した方が UDP 送信端末と競合する場合にも，若干なりとも通信性能の向上が見られ，悪影響はほとんどないことが確認できた．

7. 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 環境下ではつねに良いとは限らない．デフォルトの輻輳制御は，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低帯域であるため，ほぼつねに AP 周りがボトルネックになっている．

我々は，スマートフォ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，実機によってデータ通信の実効速度で評価し，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 を共有するすべ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，提案手法をすべての Android 端末に導入した場合にスループットとフェアネスインデックスの大きな向上が確認できた．さらには，ミドルウェアを持たない端末や UDP データを送信する端末との混在環境においても，効果があることを確認した．これは，我々の提案手法が，無線区間の端末どうしで情報交換を行い，帯域の混雑具合いに応じて輻輳ウィンドウの最大値を設定し，帯域制御をしているため，ミドルウェアを持たない端末や，UDP データの転送，また有線区間での輻輳が起きた場合には，輻輳ウィンドウは最大値以下の範囲でフレキシブルにウィンドウスケールを行うため，デフォルト以上に帯域幅を確保することはない．この輻輳制御の補正は，帯域幅を埋めるのに十分なデータ量の送出を許すため，帯域幅を埋めながらも，無駄なパケット損失や再送回数を抑えることができ，通信性能の向上に役に立つ．

本稿では，Android 端末において，この協調的通信制御手法の有効性を証明したが，ミドルウェアを組み込むことができれば，本手法はいか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，つねにすべての端末が同じ遅延時間を持っているとは限らない．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 を共有するすべての端末がつね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，メールの送信などの帯域をさほど必要としない低トラフィックなデータ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つながる．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．

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．

参考文献

[1] 熊谷菜津美，村瀬 勉，小口正人：多くの AP が近接する場合の通信品質評価，電子情報通信学会，NS 研究会，NS2012-94, pp.79–84, 京都大学 (2012 年 10 月).

[2] Wu-chun, F., Balaji, P., Baron, C., Bhuyan, L.N. and Panda, D.K.: Performance characterization of a 10-Gigabit Ethernet TOE, *Proc. IEEE Symposium on High Performance Interconnects – HOTI*, pp.58–63 (2005).

[3] Jeffrey, S.C., Gallatin, A.J. and Yocum, K.G.: End system optimizations for high-speed TCP, *Communications Magazine*, Vol.39, No.4, pp.68–74, IEEE (2001).

[4] Ha, S., Rhee, I. and Xu, L.: CUBIC: A New TCP-Friendly High-Speed TCP Variant, *Proc. SIGOPS Operating Systems Review*, Vol.42, No.5, pp.64–74 (2008).

[5] Sally, F.: TCP and explicit congestion notification, *ACM SIGCOMM Computer Communication Review*, Vol.24, No.5, pp.8–23 (1994).

[6] Hari, B. and Katz, R.H.: Explicit loss notification and wireless web performance, *Proc. IEEE Globecom Internet Mini-Conference* (1998).

[7] Victor, F. and Borden, M.: A study of active queue management for congestion control, *Proc. IEEE International Conference on Computer Communications* (*INFOCOM*), Vol.3, pp.1435–1444 (2000).

[8] Sally, F. and Jacobson, V.: Random early detection gateways for congestion avoidance, *IEEE/ACM Trans. Networking – TON*, Vol.1, No.4, pp.397–413 (1993).

[9] Ajay, B. and Badrinath, B.R.: I-TCP: Indirect TCP for mobile hosts, *Proc. 15th International Conference on Distributed Computing Systems*, IEEE (1995).

[10] Rendon, J., Casadevall, F. and Serarols, D.: Snoop TCP performance over GPRS, *IEEE VTS 53rd. Vehicular Technology Conference*, *VTC 2001 Spring*, Vol.3, IEEE (2001).

[11] 平井敏之，大坐畠智，川島幸之助：リアルタイム通信の品質改善のための無線 LAN 端末における TCP 輻輳制御（トラヒック，輻輳制御，トラヒック，NW 評価，性能，リソース管理・制御，トラヒックエンジニアリング，NW 信頼性・レジリエンシ，一般），電子情報通信学会技術研究報告 NS，ネットワークシステム，Vol.110, No.4, pp.11–16 (2010).

[12] 井尻恵也，大坐畠智，川島幸之助：CQ2010-31 TCP ウィンドウのクロスレイヤ制御による無線 LAN の通信品質改善（一般，映像配信・コミュニケーションを支える品質技術，一般），電子情報通信学会技術研究報告 CQ，コミュニケーションクオリティ，Vol.110, No.118, pp.81–86 (2010).

[13] Lawrence, S.B., O'malley, S.W. and Peterson, L.L.: TCP Vegas: New techniques for congestion detection and avoidance, Vol.24, No.4, ACM (1994).

[14] Richard J.L., Walrand, J. and Anantharam, V.: Issues in TCP vegas, Electronics Research Laboratory, College of Engineering, University of California (1999).

[15] 兼子和巳，甲藤二郎：高速回線のための TCP 輻輳制御方式（TCP-Fusion）の提案，電子情報通信学会技術研究報告 IN，情報ネットワーク，105.628, pp.207–212 (2006).

[16] 橋本匡史，長谷川剛，村田正幸：無線 LAN 環境における TCP フロー間の公平性改善手法の提案とその評価，CQ 研究会第 6 回 QoS ワークショップ (2008 年 12 月).

[17] Song, K.T.J., Zhang, Q. and Sridharan, M.: Compound TCP: A scalable and TCP-friendly congestion control for high-speed networks, *Proc. Protocols for Future, Large-Scale & Diverse Network Transports* (*PFLDNeT*) (2006).

[18] 五十嵐健，山崎憲一：広帯域無線アクセスむけデータ送信量自動調整方法の提案，インターネットコンファレンス 2007 論文集，インターネットコンファレンス実行委員会，pp.69–78 (2007).

[19] Hirai, H., Miki, K., Yamaguchi, S. and Oguchi, M.: A Study on Transmission-Control Middleware on an Android Terminal in a WLAN Environment, *Proc. 4th International Conference on Wireless, Mobile Networks & Applications* (*WiMoA-2012*), Delhi, India, pp.469–479 (May 2012).

[20] 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，マルチメディア，分散，協調とモバイル（DICOMO2012）シンポジウム，山代温泉ホテル百万石，1B-4，pp.68–75 (2012 年 7 月).

[21] Nandita, D., Refice, T., Cheng, Y., Chu, J., Herbert, T., Agarwal, A., Jain, A. and Sutin, N.: An argument for increasing TCP's initial congestion window, *ACM SIGCOMM Computer Communication Review*, Vol.40, No.3, pp.27–33 (2010).

[22] Mark, A., Hayes, C. and Ostermann, S.: An evaluation of TCP with larger initial windows, *ACM SIGCOMM Computer Communication Review*, Vol.28, No.3, pp.41–52 (1998).

[23] Android open source project, available from ⟨http://source.android.com⟩.

[24] The Linux Kernel Archives, available from ⟨http://www.kernel.org/⟩.

[25] Linaro: open source software for ARM SoCs, available from ⟨http://www.linaro.org/⟩.

[26] Sourcery G++ Lite for ARM GNU / Linux, available from ⟨http://www.codesourcery.com⟩.

[27] Iperf For Android Project in Distributed Systems, available from ⟨http://www.cs.technion.ac.il/~sakogan/DSL/2011/projects/iperf/index.html⟩.

[28] The dummynet project, available from ⟨http://info.iet.unipi.it/~luigi/dummynet⟩.

[29] Stevens, W.R.: TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms, The Internet Society (RFC2001) (1997).

[30] George, X., Polyzos, G.C., Mahonen, P. and Saaranen, M.: TCP performance issues over wireless links, *Communications Magazine*, Vol.39, No.4, pp.52–58, IEEE (2001).

[31] Jeffrey, De B., Joseph, W., Verloock, L. and Martens, L.: Evaluation of link performance of an indoor 802.11 g network, *Proc. Consumer Communications and Networking Conference* (*CCNC*), *5th IEEE*, pp.425–429, IEEE (2008).

[32] Chiu, D.-M. and Jain, R.: Analysis of the increase and decrease algorithms for congestion avoidance in computer networks, *Computer Networks and ISDN Systems*, Vol.17, pp.1–14 (1989).
[33] 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 通信時の通信性能の考察，*DEIM2011*，B6-1，伊豆修善寺 (2011 年 3 月).

平井 弘実 （学生会員）

1988 年生．2011 年お茶の水女子大学理学部情報科学科卒業．2013 年同大学大学院修士課程修了．同年日本電信電話株式会社入社．将来ネットワークの研究に従事．電子情報通信学会会員．

山口 実靖 （正会員）

1974 年生．2002 年東京大学大学院工学系研究科電子情報工学専攻博士課程修了．博士（工学）．同年より東京大学生産技術研究所研究員．2006 年より工学院大学工学部情報通信工学科講師．2007 年より同大学同学科准教授．ネットワークコンピューティングの研究に従事．電子情報通信学会会員．

小口 正人 （正会員）

1967 年生．1990 年慶応義塾大学理工学部電気工学科卒業．1992 年東京大学大学院修士課程修了．1995 年同大学院博士課程修了．工学博士．学術情報センター中核的研究機関研究員，東京大学生産技術研究所特別研究員，中央大学研究開発機構助教授，お茶の水女子大学理学部情報科学科助教授を経て，2006 年より同教授．ネットワークコンピューティング・ミドルウェアに関する研究に従事．IEEE，ACM，電子情報通信学会各会員．