DEIM Forum2013 E6-1

無線LAN環境における 環境情報に基づいたTCPパラメータ設定手法の評価

飯尾　明日香† 　　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
E-mail: †asuka-i@ogl.is.ocha.ac.jp, ††oguchi@computer.org

あらまし　近年，使用用途に応じた様々なセンサの開発が進んでおり，自動車など無数の高性能センサが搭載された端末では，精密な周囲の環境情報を取得可能である．このセンサ情報から生成される Conext は，自動車の場合には，主に事故防止やドライバの安全運転支援のため車両制御に利用されている．

一方で，WLAN(Wireless Local Area Network) 通信を行うそれらの端末における通信パラメータは周囲の状況に応じた設定がなされていないため，リソースを活用した効率的な通信を行っているとは言えない．

そこで，本研究では，上位層である TCP(Transmission Control Protcol) の輻輳制御に着目し，ここに環境情報として周辺端末数を利用することで，通信性能を改善する手法を検討する．通常，TCP の輻輳ウィンドウサイズ（cwnd）は周辺の端末数に関わらず輻輳制御アルゴリズムにより動的に制御されているが，本稿では，各端末が Context に応じ cwnd を適切な値に固定して通信を行う独自の TCP を提案し，これを用いて通信を行った場合のスループットをシミュレーションにより測定して，評価を行った．

キーワード　環境情報，Context，TCP，輻輳制御方式，WLAN 通信

A Evaluation of TCP Parameter Setup Technique Based on Environmental Information in Wireless LAN.

Asuka IIO† and Masato OGUCHI†

† Ochanomizu University　　2-1-1 Otsuka, Bunkyou-ku, Tokyo, 112-8610, JAPAN
E-mail: †asuka-i@ogl.is.ocha.ac.jp, ††oguchi@computer.org

1. はじめに

現在，自動車には数百種類のセンサが搭載されており，自動車はこれらのセンサから GPS（全地球測位システム）による車両位置を始め，車両間隔，スピード，周辺端末数，加速度など数多くの情報を取得できる．また，車載センサはドライバの安全機能強化の面から高精度化が進んでおり，搭載数も年々増加の傾向にある．このようなセンサから得た周囲の環境情報を Context と呼ぶが，この Context は，自動車の場合には事故防止や車両安定制御等に役立てられている．

一方，近年 WLAN（Wireless Local Area Network）による通信が一般化しているが，WLAN において通信に用いられる通信パラメータは各々が独自のアルゴリズムに従って設定されており，その場の環境に応じた通信設定は行われない．したがって，センサから環境情報を取得可能な端末であってもリソースを効率的に利用した通信が行われておらず，必ずしも最適ではない状態で通信を行っているという問題がある．そこで，本研究では，環境情報を無線通信パラメータ設定に利用し，各端末が周囲の状況に応じた最適な設定を行うことで，通信効率を向上させる手法の検討を行う．

本稿では，環境情報の利用先として上位層の TCP（Transmission Control Protocol）に着目した．TCP において，確認応答（ACK）を受けずに送信できるパケットの最大数である輻輳ウィンドウサイズ（cwnd）は輻輳制御アルゴリズムによりエンドツーエンドで制御されているため，各端末は必ずしも周囲の状況に合った通信を行っているとは言えない．そこで，Context として周辺端末数を利用し，各端末が最適な cwnd を用いて通信を行うことで通信性能の向上を目指す．

本研究では，輻輳制御アルゴリズムにより cwnd が制御される一般的な TCP を用いて通信を行った場合と，周辺端末数に応じ最適な cwnd 値に固定して通信を行う独自の TCP を用いて通信を行った場合のそれぞれにおいて，周辺端末数に伴

うスループットの変化をシミュレーションにより測定し，評価を行った．なお，実験にはネットワークシミュレータとして OMNeT++ version4.1[1] を使用し，シミュレーションモデルは INET Framework[2] および INETMANET[3] を使用した．

2. Context

2.1 Context とは

Context は「実体を特徴づけることのできるあらゆる情報」と定義されている．ここでの実体とは，ユーザとアプリケーションの間のやり取りに関連していると考えられる人，場所，物のことを指しており，これにはユーザやアプリケーション自体も含まれている．Context をこのように定義することで，アプリケーションを開発する際に必要となる，実体のその時の状況がわかるような情報の列挙がより容易になる．[4][5]

図 1 は，センサ情報の取得から Context 生成の流れを表している．無数のセンサから取得した環境情報を収集し，さまざまなアプリケーションで利用しやすいように抽象化することで，より正確な Context を生成している．図 1 のように，Context はデータベースで管理を行う．

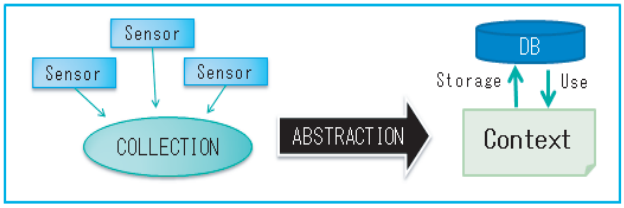

図 1　センサ情報から Context を生成

Context はさまざまなアプリケーションで利用することができることから，通信性能の改善にも利用可能であると考えられる．

2.2 Context-Aware Computing

Context-Aware とは，たとえ人が意識しなくとも，コンピュータがリアルタイムで実体の情報を収集・処理し，Context 化して利用する技術や概念のことを示している．ユーザに必要な情報やサービスを提供するために Context を使用するシステムを Context-Aware であると呼ぶ．

Context-Awere Computing は，Context を利用して企業や個人の情報利用環境を改善しようとする取り組みで，システムやユーザがその時に必要としている情報や機能を判断し，ユーザにサービスとして提供するものである．つまり，様々なシステムやセンサなどから得られる実体の情報，すなわち，Context と人とをリアルタイムで結びつける技術である．これは多数の技術の連携によって実現するが，近年，システム連携の仕組みが発展してきたことから Context-Awere Computing は実現が期待され，重要性が高まっている．

2.3 関 連 研 究

2.3.1 C2D2 Engine

Context 利用に関連する研究として文献 [6] に示した C2D2(Context-aware Collection，Decision，and Distribution) Engine について説明する．

無線通信が Context に強く依存することは理解されているが，一方で，通信性能を改善するために Context を利用するようなプロトコルがまだ完全には開発されていない．そこで，ネットワークスタックの多層にわたり既存の通信プロトコルと Context を接続する Context-aware のフレームワークの基礎を築いたという研究である．

車通信では，動的なチャンネル変更や端末の高い移動性が考えられるため，接続状態を維持でき，高スループットでの通信が可能であるような環境が望まれる．そこで，この研究では，アプリケーションが車通信における QoS を満たす上で利用可能な Context を判断し，生成されたデータをネットワークスタックの各層へ振り分けるフレームワークについて述べている．Context は図 2 に示すように Collection，Decision，Distribution を経由し，それぞれが適合する問題へ与えられる．すなわち，Decision Engine がセンサを用いて収集したデータの中から無線通信性能の改善に利用できるものを選択し，選択したデータを無線通信プロトコル上の適合する層へ分配するシステムである．

本研究はこのうち，TCP/UDP 層における Context-aware Decision およびその Context の TCP/UDP 層への適用において利用可能な技術の確立を目指したものであるといえる．

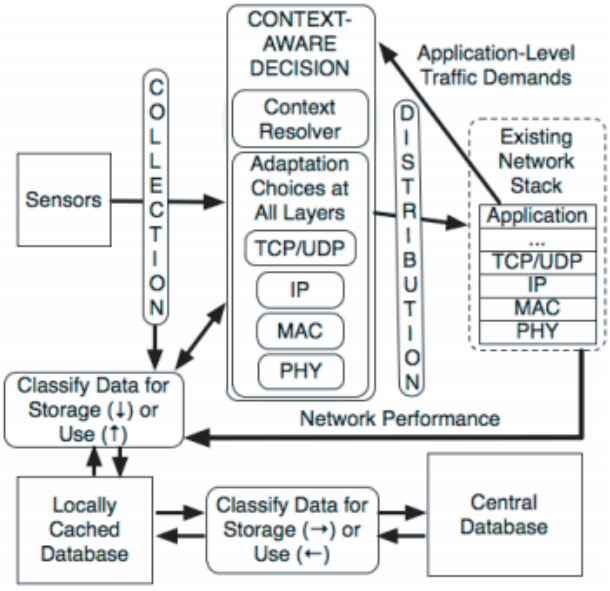

図 2　C2D2 システムのダイアグラム

2.3.2 MAC 層への Context 利用

無線 LAN 通信への Context 利用の有用性は，文献 [7] にも示されている．これは，MAC 層の CSMA/CA(Carrier Sense Multiple Access with Collision Avoidance) のバックオフ制御に着目し，Contention Window 初期値の設定に，周辺端末数を利用し，通信リソースの効率的な利用を目的とする研究である．実環境に近い環境を想定して条件設定を行い，シミュレーションによる評価を行うことにより，MAC 層における環境情報利用の有用性が示されている．

本研究では，通信における MAC 層の上位層である TCP 層への Context 利用を考える．

3. TCP

3.1 TCP による輻輳制御

WLAN の国際標準として IEEE 802.11 が規定されており，IEEE 802.11 の上位層においては TCP により輻輳制御が実現されている．輻輳とはネットワークの混雑を表す言葉であり，ネットワーク上で多量のトラフィックが発生して，通常の通信が困難になることをいう．TCP は，輻輳制御により輻輳を回避する仕様となっている．代表的な TCP として Tahoe，Reno，NewReno，Vegas，Cubic 等があり，輻輳制御アルゴリズムは TCP によって様々な方式となっている．

TCP では，ウィンドウフロー制御をすることによって，輻輳の防止・早期回避を可能にしているが，一方で，輻輳回避フェーズにおける輻輳による cwnd の減少幅が，その後の線形増加時の増加幅と比較し大きいため，一度輻輳が起き cwnd が下がると，cwnd が増加するまでに時間がかかり，輻輳制御が原因となったスループットの低下が起こる場合がある．以下では，TCP Tahoe と TCP Reno の輻輳アルゴリズムについて説明し，本稿で比較評価に用いた TCP である TCP NewReno について触れる．

3.2 TCP Tahoe

TCP Tahoe は，スロースタートフェーズと呼ばれる，ACK セグメントを受信した分だけ cwnd を増加させる動作を行う (式 (1)，if 条件式).

$$cwnd \longleftarrow \begin{cases} cwnd + 1 & \text{if cwnd} < \text{ssthresh} \\ cwnd + \frac{1}{cwnd} & \text{otherwise} \end{cases} \quad (1)$$

cwnd=1 でセグメントを送信すると，受信側から 1 つの ACK セグメントが送信されるため cwnd は 2 となる．cwnd が 2 のとき 2 つのセグメントを送信するため，受信側からは 2 つの ACK セグメントが送信され，cwnd は 4 となる．このようにスロースタートフェーズでは cwnd を 1,2,4,8,16,…と指数関数的に増加させていく．ここで，ssthresh はスロースタートから輻輳回避のフェーズへ移行する際の閾値である．ssthresh 初期値は多くの実装で広告ウィンドウサイズが設定されるが，ある程度大きい値であれば任意である．

また，輻輳が起こると ssthresh は以下のように更新される．

$$ssthresh \longleftarrow \frac{ssthresh}{2} \quad (2)$$

その後，cwnd を 1 まで下げ，再びスロースタートフェーズを行う．やがて cwnd が ssthreth に達すると輻輳回避フェーズへ移行し，cwnd を式 (1) の otherwise 条件式を用いて線形的に増加させる．ここで再度輻輳が起こると，再び式 (2) を用いて ssthreth を更新し，cwnd を 1 まで下げ，スロースタートフェーズへ移行する．TCP Tahoe はこの動作を繰り返す（図 3）．また，高速再送アルゴリズムが採用された点も TCP Tahoe の特徴である．

3.3 TCP Reno

TCP Reno は TCP Tahoe がベースのアルゴリズムで，TCP Tahoe に対してエラーが起きても必ずしも cwnd を 1 まで下げないように改良したものである．すなわち，始めはスロースタートフェーズで cwnd の急速な増加を図り，その後，cwnd が ssthresh に到達すると，輻輳回避フェーズへと移行する．輻輳回避フェーズでは，輻輳が起こると ssthresh を半分に更新し，cwnd を ssthresh まで下げてから線形的に増加させ，輻輳が起こる度にこれを繰り返す (図 4)．また，パケットロスなどによりタイムアウトが起きると，cwnd を 1 に変更しスロースタートフェーズの動作へと戻る．

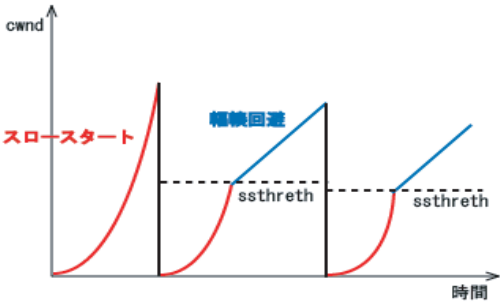

図 3　TCP Tahoe における cwnd の変化

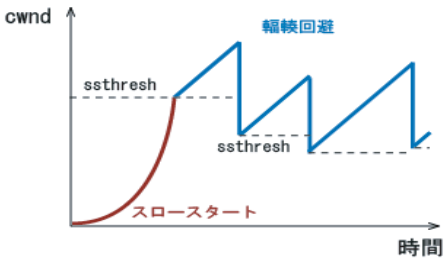

図 4　TCP Reno における cwnd の変化

3.4 TCP NewReno

TCP Reno では SACK（selective acknowledgement）がない場合に，複数パケットが損失すると高速再転送および高速リカバリアルゴリズムが有効に動作しない問題があった．この高速リカバリアルゴリズムを修整したものが TCP NewReno である．

4. 独自の TCP

3. 節で述べたように，通常，TCP では cwnd を輻輳制御アルゴリズムにより動的に制御している．しかし，本研究では Context として周辺端末数を利用し，各端末が初めから適切な cwnd 値に固定して通信を行うことで，通信性能の向上を試みる．そこで，文献 [8] より，帯域幅遅延積（式（3）上式）を用い，これを端末数に応じて平等に分けることで，公平かつ効率の良い通信を目指す．

$$\begin{cases} \text{帯域幅遅延積 } [bit] = \text{帯域幅 } [bit/s] \times RTT[s] \\ cwnd = \text{帯域幅遅延積} \div \text{端末数} \end{cases} \quad (3)$$

ここで，RTT（Round Trip Time）は往復遅延時間を表す．

本稿では，すべての端末数において式（3）を適用し，端末

数ごとに適切な cwnd を定めて通信を行う TCP を独自の TCP と呼ぶ.

5. 検 証 実 験

5.1 シンプルな環境

初めに，AP（Access Point）から等距離にある複数の端末が，AP を経由しサーバにパケットを送信するシンプルなシミュレーションモデルを使用した (図 5).

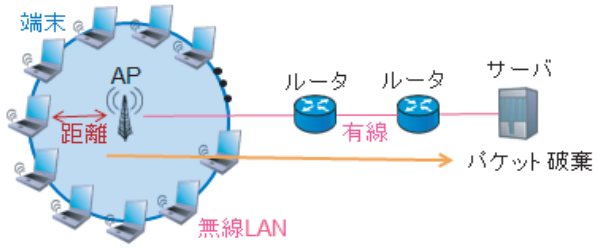

図 5　シミュレーションモデル

このモデルにおいて，全端末数を用い 4. 節で述べた式（3）を適用し cwnd を定めて通信する独自の TCP と，その比較対象として一般的に使用されている TCP である TCP NewReno の 2 種類の TCP を使用して，端末数に伴うトータルスループットを測定した．端末は，AP を中心とした円周上に均等に配置しているものとする．また，無線規格は IEEE802.11g を使用し，広告ウィンドウサイズは十分大きい値に設定した．

シミュレーションパラメータを表 1 に示す．なお，ここでの RTT とは，OMNeT++上で設定するコネクションの往復遅延時間のみを表した時間であり，実際の測定値とは異なる．

このモデルを使用し，まず，AP と端末間の距離 50m 時のトータルスループットを測定し，次に，距離 100m 時のトータルスループットを測定した．このとき，独自の TCP で式（3）に用いる RTT は，AP からの距離 50m 時では 200msec とし，100m 時では 180msec とした．これは，実験を行った際の RTT の増減幅内，または，増減幅の前後で，それぞれ高いスループットが求められた数値である．

シミュレーション結果を図 6，図 8 に示す．また，それぞれの場合における各端末の RTT の増減を図 7，図 9 に示す．これは，縦軸が RTT[sec]，横軸がシミュレーション時間であり，端末数を 20 台とした場合の結果である．

表 1　シミュレーションパラメータ

無線規格	IEEE8002.11g
シミュレーション時間	20sec
データレート	54Mbps
RTT	40msec
周波帯域	2.4GHz
最大セグメントサイズ	65,280Byte
広告ウィンドウサイズ	33,423,360Byte

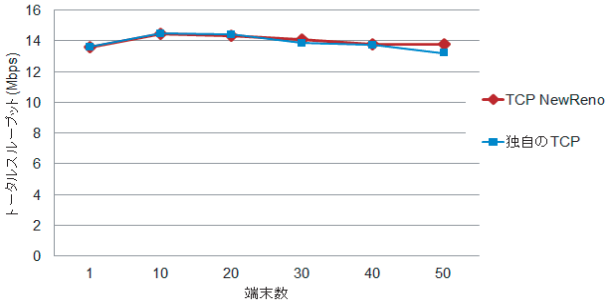

図 6　距離 50m 時のトータルスループット比較

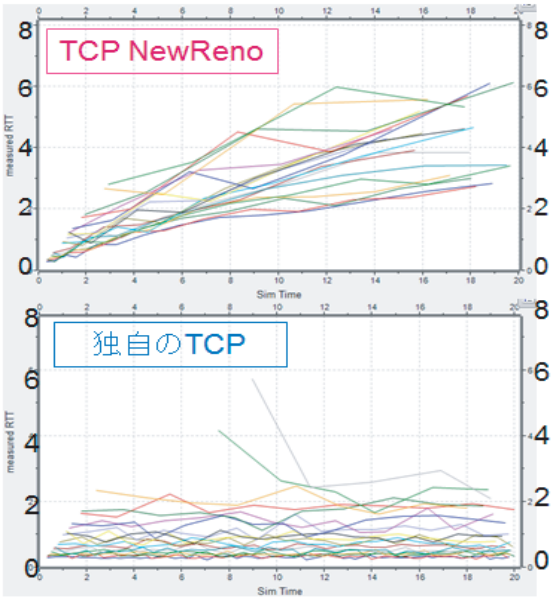

図 7　距離 50m 時の RTT 比較

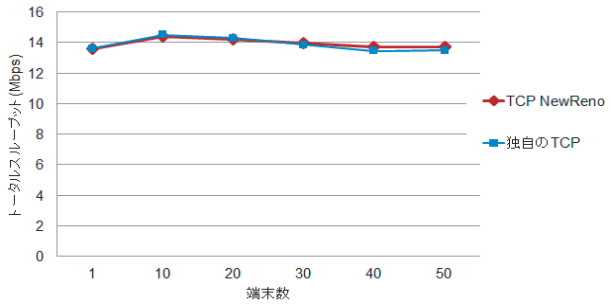

図 8　距離 100m 時のトータルスループット比較

5.2 実環境に近づけた環境

先ほど実験を行ったモデルでは，端末を AP から特定の距離に均等に配置していた。このような環境は，実環境では考えにくい。そこで，より実環境に近づけるため，端末を 2 グループに分け，AP と端末間の距離を変え，それぞれ均等に配置するモデルを構築し，実験を行った（図 10）.

5.2.1 AP と端末間の距離 25 mと 50 mの共存

AP と端末間の距離が 25 m（距離 I）の端末と，AP と端末間の距離が 50 m（距離 II）の端末を同数用意し，5.1 節と同様に，AP を中心とした円周上に端末を均等に配置したモデルを考えた（図 10）．シミュレーションパラメータは表 1 と同様で

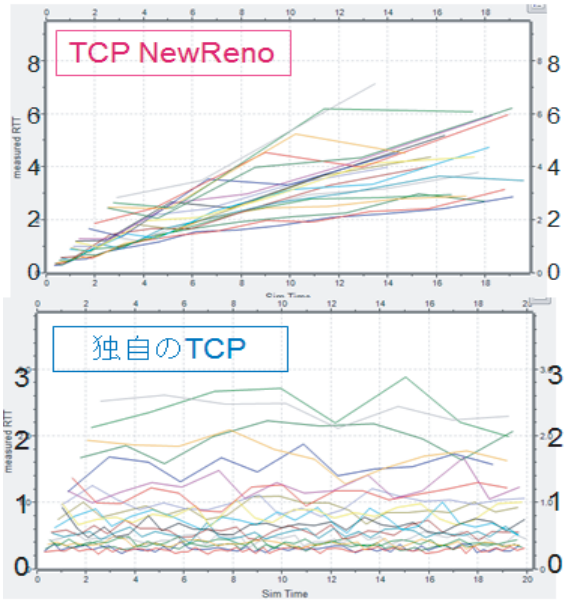

図 9　距離 100m 時の RTT 比較

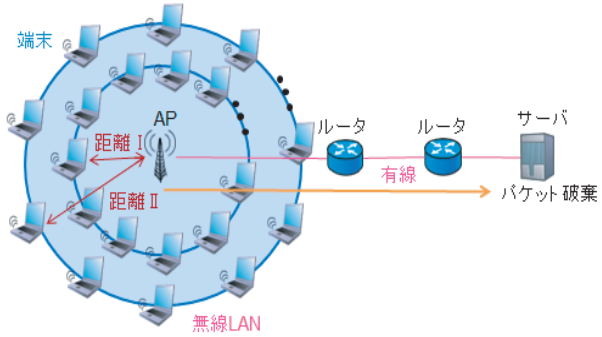

図 10　シミュレーションモデル

ある．また，独自の TCP で式（3）に用いる RTT は，帯域に若干の余裕を確保できるように，パラメータに設定した 40msec よりも少し小さい 30msec とし，cwnd を決定した．

また，シミュレーション結果の図 11 における横軸の端末数とは，図 10 において距離 I に配置した端末数と距離 II に配置した端末数の合計値である．また，図 12 は各端末の RTT の増減を示す．

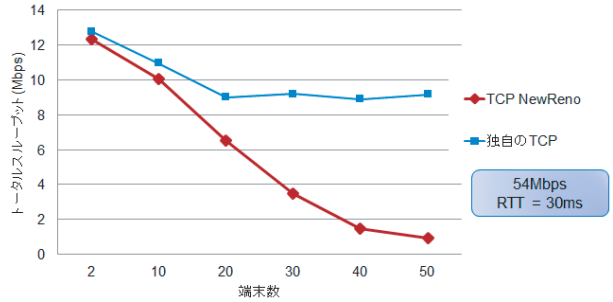

図 11　距離 25m，50m 共存時のトータルスループット比較

5.2.2　AP と端末間の距離 50 mと 100 mの共存

続いて，距離 I を 50m，距離 II を 100m とし，同様に実験を行った．こちらも，独自の TCP で式（3）に用いる RTT も同様

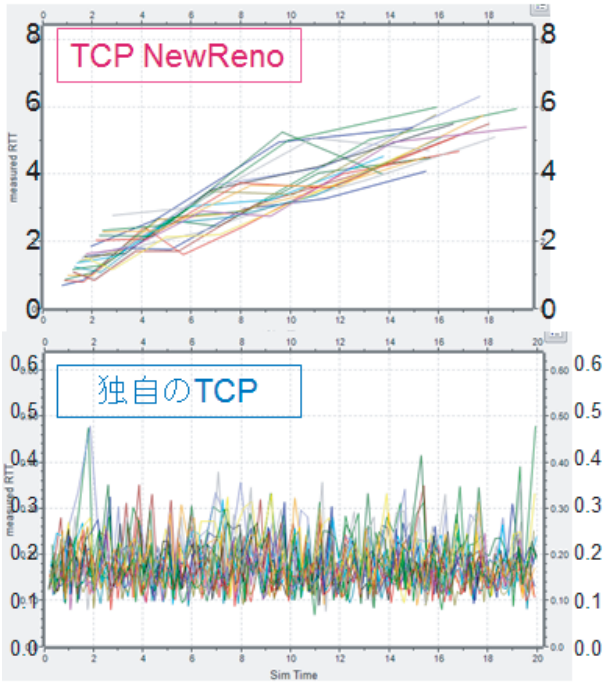

図 12　距離 25m，50m 時の RTT 比較

に，帯域に余裕を確保できるよう 30msec とし，cwnd を決定した．

シミュレーション結果のトータルスループットを図 13，RTT を図 14 に示す．

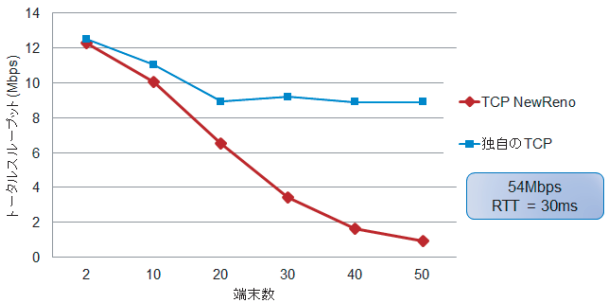

図 13　距離 50m，100m 共存時のトータルスループット比較

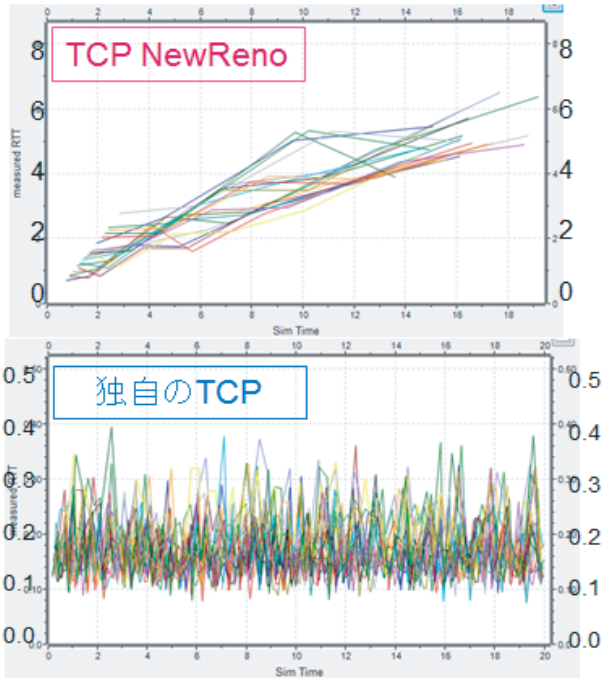

図 14　距離 50m，100m 時の RTT 比較

5.3　考　　察

図 6，図 8 では，TCP NewReno と独自の TCP の 2 種類の通信性能はほぼ変わらないと言える．これは，簡単な WLAN モデルを用いて実験を行ったことから，TCP NewReno に大きな非効率が発生しておらず，cwnd の値の違いによる性能差が出にくかったためと考えられるが，図 7，図 9 において RTT を比較してみると，どちらも TCP NewReno はシミュレーション時間の増加に伴い 6 s 程度まで増加しているのに対し，独自の TCP では 3 s 程度で安定している．よって，提案手法では従来の TCP より効率の良い通信が行われていると考えられる．

一方で，端末配置に 2 種類の距離を用いた環境では，TPC NewReno のトータルスループットは，端末数の増加と共に大きく低下しているが，独自の TCP のトータルスループットは，端末数が増加しても大きな低下は見られず，その優位性がはっきりと現れた（図 11，図 13）．また，図 12，図 14 の RTT の比較においても，その優位性を明確に示した．

より実環境に近い複雑な端末配置では TCP NewReno の輻輳制御が効率良く機能していないが，独自の TCP における cwnd 設定では効率の良い通信が行われていると考えられる．このことから，端末数に応じて帯域を均等に分けた独自の TCP の有用性が示せた．

6.　ま　と　め

WLAN 通信環境において，一般に使用されている輻輳制御を行う TCP NewReno と，Context に応じて cwnd を適切な値に保ったまま通信を行う独自の TCP を用い，端末数に伴うトータルスループットをシミュレーションにより求め，その性能を比較した．その結果，TCP NewReno では端末の配置が AP から等距離にあるというシンプルなモデルの場合，通信性能の低下が見られなかったが，端末の配置を実環境に近づけ距離を均一でなくしたモデルの場合では，端末数の増加に伴いトータルスループットが大きく低下したため，従来の TCP では効率の良い通信が行われていないことがわかった．また，独自の TCP を用いたとき，端末の配置が複雑な場合でも，TCP NewReno ほど端末数の増加による性能の低下は見られなかった．

以上から，実環境では，帯域幅遅延積を周辺端末と平等に分けて通信を行うことで，従来の TCP より良い性能での通信が可能であると考えられる．

7.　今後の課題

今後は，AP からの距離に応じてデータレートが異なる環境における提案手法の評価を行う．さらに，端末の移動性を考慮したより複雑なモデルを用いてシミュレーションを行い，通信性能の向上を実現したい．また，最終的には，周囲の端末数が大きく変化した際に，リアルタイムで通信パラメータを変更しその時の状況に適応した通信を行う TCP を実装したい．

謝　　辞

本研究を進めるにあたり，株式会社トヨタ IT 開発センターの Onur Altintas 氏，松本真紀子氏に大変有用なアドバイスをいただきました．深く感謝いたします．

文　　献

[1] OMNeT++:
http://www.omnetpp.org/
[2] INET Framework：
http://inet.omnetpp.org/
[3] INETMANET Framework for OMNEST/OMNeT++ 4.x (based on INET Framework)：
https://github.com/inetmanet/inetmanet/wiki
[4] Daniel Salder, Anind K.Dey and Gregory D.Abowd, ”The Context Toolkit : Aiding the Development of Context-Enabled Applications”, In Proccedings of CHI’99,PIttsburgh,PA,May 15-20,1999,(to appear).ACM Press.
[5] Day, Anind K., ”Understanding and Using Context”(2001). Human-Computer Interaction institute.Paper 34.
http://repository.cmu.edu/hcii/34
[6] Joseph Camp, Onur Altintas, Rama Vuyyuru, and Dinesh Rajan, ”Context-awere Collection,Decision,and Distribution(C2D2) Engine for Multi-Dimensional Adaptation in Vehicular Networks”, VANET ’11, pp.85-86, September 23, 2011
[7] 松本真紀子，小口正人，”無線端末の周辺情報を利用した MAC 層におけるマルチプルアクセス手法の一検討”，電子情報通信学会ネットワークシステム研究会，NS2011-106，pp.13-18，2011 年 11 月
[8] W.Richard Stevens 著，橘 康雄，井上 尚司訳（2000）『詳解 TCP/IP〈Vol.1〉プロトコル』ピアソンエデュケーション 327pp.