環境情報に基づく トランスポート層における通信制御手法の評価

飯尾　明日香[†1]　　小　口　正　人[†1]

現在，自動車のように無数のセンサが搭載された端末からは，その物体の状況を特徴づけるような環境情報，すなわち，Context を取得できる．センサ情報から生成された Context は，自動車においては主に事故防止やドライバの完全運転支援のため車両制御に利用されているが，これは無線通信における最適なパラメータ設定にも利用可能であると考えられる．無線通信において，一般に通信パラメータは周囲の状況に関わらず静的に設定されているものが多く，端末は必ずしも効率の良い通信設定がされているとはいえないことから，環境情報を利用することにより，通信効率の改善が期待できる．

本研究では，通信の上位層である TCP 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による通信性能の改善手法を検討する．TCP の輻輳制御において性能を決定する重要なパラメータである輻輳ウィンドウサイズは，周辺の端末数等に関わらず，エンド間の情報のやり取りのみに基づく輻輳制御アルゴリズムにより制御されているため，各端末は周囲の状況を考慮した最適な通信を行っているとは言えない．本稿では，輻輳制御を行う通常の TCP と，輻輳ウィンドウサイズを一定値に保つ TCP を，通信シミュレーションにより性能比較することにより，環境情報に基づく通信制御手法を確立するための基礎検討を行った．

An Evaluation of Communication Control by Using the Context on Transport Layer

Asuka Iio[†1] and Masato Oguchi[†1]

In recent years, intelligent services that consider surrounding conditions based on sensor information have been studied extensively. On the other hand, many parameters of wireless communication are uniquely configured, regardless of the communication environment.

In this paper, we have proposed to configure parameters using the context for improving wireless communication efficiency. We have focused on congestion control algorithm on Transport Layer, and fixed congestion window size (cwnd) to certain fixed value properly using it. Although the congestion window is controlled by the congestion algorithm on the conventional TCP, it may not be a setup suitable for the surrounding environment. Hence we have evaluated communication performance by deciding cwnd under scenarios which are closer to real environment by simulations.

1. はじめに

近年，多種多様で高性能なセンサが数多く普及しており，これらのセンサ情報を組み合わせることでコンピュータは人や端末の置かれている状況を理解できるようになってきている．このようにセンサから得た周囲の環境情報を Context と呼ぶが，Context はさまざまアプリケーションにおいて制御に利用されている．コンテキストアウェアネスはその代表例であり，Context をもとに，ユーザの置かれている状況を判断，理解し，ユーザに適切なサービスを提供するものである[1)2)]．

一方，近年 Wireless Local Area Network（WLAN）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各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資源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[3)4)]．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適した設定を行うことで，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 Transmission Control Protocol（TCP）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（cwnd）サイズは輻輳制御アルゴリズムにより制御されているため，各端末は必ずしも周辺の状況に合った通信を行っているとは言えない．そこで，輻輳制御アルゴリズムにより cwnd サイズが制御される一般的な TCP を用いて通信を行った場合に対し，cwnd サイズを一定値とする TCP を用いて通信を行った場合において，端末数や往復遅延時間（RTT）に応じたスループットの変化をシミュレーションにより測定し，性能を比較する．これにより，環境情報に基づき cwnd サイズの最適値を決定する制御の基礎評価を行う．

†1 お茶の水女子大学
Ochanomizu University

2. Context-Aware

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，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，これにはユーザとアプリケーション自体も含んでいる．Context をこのように定義することで，ある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環境情報の列挙がより容易になる[1)2)]．

図 1 は，センサ情報の取得から Context 生成を行う際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集め，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により Context は生成され，データベースで管理されている．

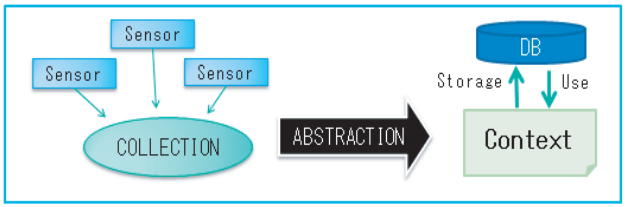

図 1 センサ情報から Context を生成

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

ユーザに対し，関連する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．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．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情報，すなわち，Context と人とをリアルタイムで結びつけ，ユーザがその時に必要な情報や機能を判断して，サービスとして提供する技術である．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-Aware Computing は実現が期待され，重要性が高まっている．

これまで Context-Aware Computing は，ユーザの嗜好判断などに用いられることを検討されてきた．本研究においては，これを通信システムの性能改善に応用する事を試みている．

2.3 関 連 研 究

Context 利用に関連する研究として参考文献 [3] に示した Context-aware Collection，Decision，and Distribution(C2D2) Engine について説明する．

無線通信が Context に非常に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．そこでこの研究においては，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 との接続を検討し，Context-aware のフレームワークの基礎を築いた．

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．

3. TCP における輻輳制御

3.1 TCP 輻輳制御アルゴリズムの概要

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 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．

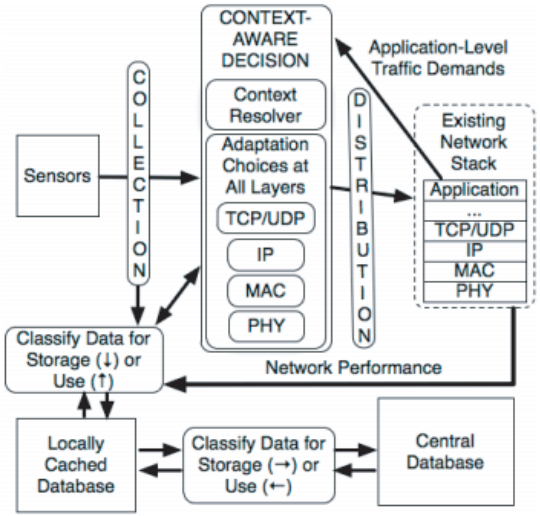

図 2 C2D2 システムのダイアグラム

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 サイズの減少幅が，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 サイズが下がると，cwnd サイズ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，本稿で比較評価に用いた TCP である TCP Tahoe と TCP Reno の輻輳制御アルゴリズムについて説明する．

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 サイズを増加させる動作を行う (式 (1)，if 条件式)．

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

cwnd サイズ 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 サイズは 2 となる．cwnd サイズが 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 サイズは 4 となる．このようにスロースタートフェーズでは cwnd サイズを 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とする．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．

$$ssthresh \longleftarrow \frac{ssthresh}{2} \tag{2}$$

その後，cwnd サイズを 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 サイズが ssthreth に達すると輻輳回避フェーズへ移行し，cwnd サイズを式 (1)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 サイズを 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．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．TCP Tahoe における cwnd の変化の様子を図 3 に示す．

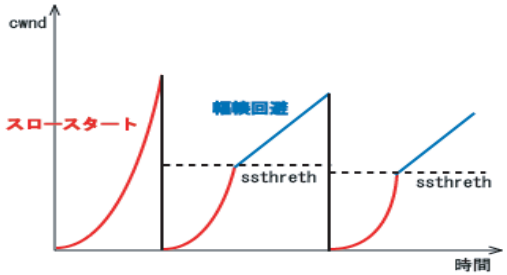

図 3 TCP Tahoe における cwnd の変化

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 サイズを 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 サイズの急速な増加を図り，その後，cwnd サイズが 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 サイズを 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 サイズを 1 に変更しスロースタートフェーズの動作へと戻る．

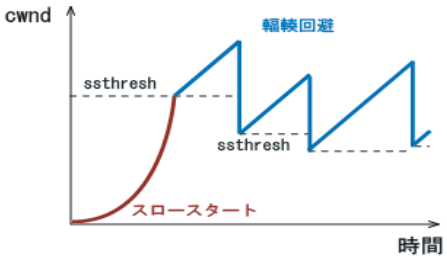

図 4　TCP Reno における cwnd の変化

3.4　TCP NewReno

TCP Reno では selective acknowledgement（SACK）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.　ネットワークシミュレータ

本研究ではネットワークシミュレータとして OMNeT++を使用している．OMNeT++は，Andras Varga によって開発されたオープンソースでオブジェクト指向の離散イベント型のネットワークシミュレーションフレームワークで，対応 OS は Unix および Windows である[5]．

本研究において，シミュレーションモデルは OMNeT++上で動作する TCP/IP のシミュレーションパッケージである INET Framework を拡張した INETMANET を使用した．INET Framework は，UDP, TCP, SCTP, IP, IPv6, Ethernet, PPP, 802.11 などさまざまなプロトコルを OMNeT++上で動作させることができるシミュレーションモデルである．これに対し，INETMANET はアドホックネットワークなど INET Framework に含まれていない無線プロトコルに対応した仕様となっている．

5.　検 証 実 験

5.1　概　　要

本稿では，複数の端末 (クライアント) と 1 台のサーバが通信を行っており，クライアントから送信されたパケットが，サーバ側で Sink(パケット破棄) されるシナリオを想定した (図 5)．このとき各クライアントは，WLAN でアクセスポイント (AP) に接続されており，AP から有線でルータを 2 台経由しサーバへパケットを送信している．

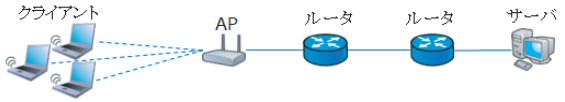

図 5　実験に用いたモデル

ここで，輻輳制御が行われている TCP（TCP Reno，TCP NewReno）と，輻輳制御が行われない TCP（No Congestion Control）の 3 種類の輻輳制御モデルを用い，それぞれのクライアント数に応じたスループットを測定した．TCP Reno，TCP NewReno において，cwnd サイズは式 (1) および (2) が適用され，各々のアルゴリズムに従い動的に更新されながら通信が行われる．一方，No Congestion Control における cwnd サイズは通信の初めからこちらで指定した一定値を保つ設定となっている．このシナリオでは，無線規格として IEEE 802.11 を使用し，その他のパラメータは表 1 のように設定しシミュレーションを行った．結果を図 6，図 7 に示す．

なお，図 6 の実験の際，cwnd は最大セグメントサイズ（mss）と同じ値と，それを 2 倍した mss より少し大きな値，さらに，INETMANET で設定できる最大の値の 3 種類を用いて TCPReno と性能を比較した．このとき，RTT は 200msec に固定し，クライアント数は 1〜50 とした．

表 1 シミュレーションパラメータ

Simulation Time	40sec
Advertised Window	33,423kByte
Data Rate	6Mbps
Carrier Frequency	2.4GHz
Maximum Segment Size	65,280Byte
Maximum Sending Power	20mW

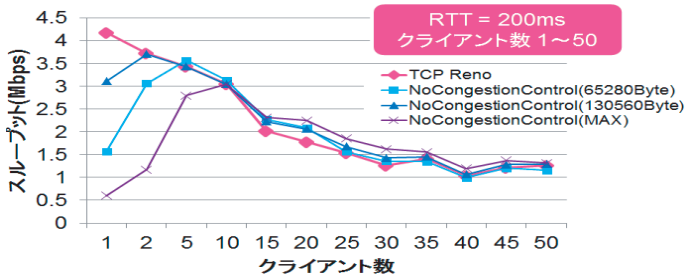

図 6 クライアント数に伴うトータルスループット

図 6 から，RTT が 200ms の場合においてクライアント数が 10 以下では TCPReno を利用した通信が良いことがわかる．一方で，クライアント数が 10～35 となると，TCPReno よりも適した cwnd サイズが存在するのではないかと考えられる．しかし，ここで設定した cwnd サイズが適当であるとは限らないため，図 7 および図 8 のようにクライアント数を固定し cwnd サイズを変化させた場合のスループットを測定した．

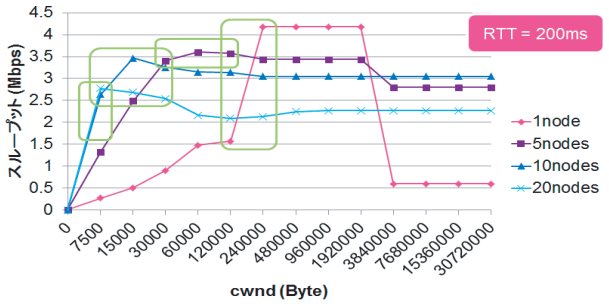

図 7 cwnd サイズに伴うトータルスループット

ここで，図 7 をもとに，それぞれのクライアント数ごとに，スループットが大きくなると考えられる cwnd サイズの範囲を詳細に測定したグラフを示す．

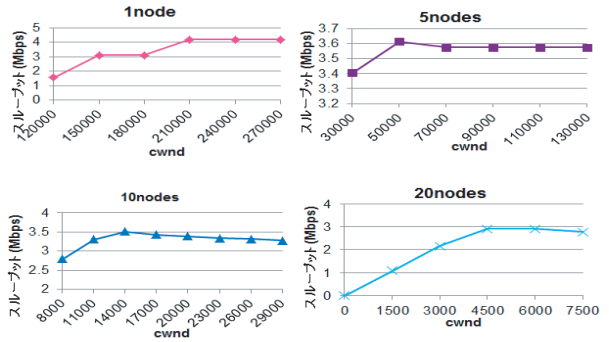

図 8 cwnd サイズに伴うトータルスループット 2

次に，クライアント数 10 と 20 において図 8 から得られた最適と考えられる cwnd を用いて通信を行った際の通信性能と，cwnd サイズを mss に設定した TCP と，TCPNewReno と性能を比較した結果を示す．

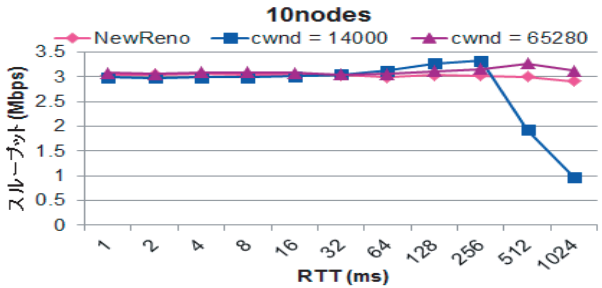

図 9 クライアント数 10 の場合のスループット比較

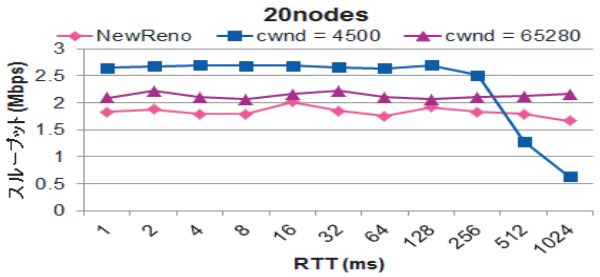

図 10 クライアント数 20 の場合のスループット比較

5.2 考　　察

シミュレーション結果から，クライアント数が 10 以下の場合は従来の TCP でも周囲の状況に適した通信が十分行われていると言える．しかし，RTT が 1〜256msec 程度までの環境において，クライアント数が 10 よりも大きい場合には，クライアント数に応じた cwnd サイズを設定することで従来の TCP を利用した通信よりも性能の向上が見込めることがわかった．

このことから，自動車における通信のように，ある程度多くのクライアント数が想定される通信環境においては，従来の TCP 輻輳制御アルゴリズムが必ずしも効率の良い通信を行っておらず，最適な cwnd サイズが他に存在すると言える．この値はクライアント数に依存すると考えられることから，Context を利用すれば，各々の通信環境に応じて従来の輻輳制御より適した cwnd サイズを決めることができる．すなわち Context をベースとした輻輳制御方式が確立できる可能性があると考えられる．

6. おわりに

WLAN において遅延環境モデルを用い，一般に使用されている輻輳制御を行う TCP Reno，TCP NewReno と，cwnd サイズを一定に保つ TCP である NoCongestionControl のそれぞれのクライアント数に伴うスループットをシミュレーションにより求めた．この結果から，本稿の場合クライアント数が 10 よりも多い場合に従来の TCP では最適な通信が行われていないことがわかった．

本研究の結果から，周辺端末数を Context として cwnd サイズの設定方法を改変することで端末数が大きい場合の通信性能を向上できると考えられるため，今後は，さまざまな端末数においてシミュレーションを行い，それぞれに最適な cwnd サイズを求めたい．また，本稿ではトータルスループットでの性能評価を行ったが，周囲の端末との公平性を重視した最適な通信設定の手法も検討していきたい．

謝　　辞

本研修を進めるにあたり，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．

参 考 文 献

1) 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.
2) Day,Anind K., ”Understanding and Using Context”(2001). Human-Computer Interaction institute.Paper 34.
http://repository.cmu.edu/hcii/34
3) 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 2011.
4) 松本真紀子，Onur Altintas，西堀満洋，小口正人，”環境情報を利用したマルチプルアクセス手法の一検討”，情報処理学会第 73 回全国大会，1V-6，2011 年 3 月．
5) OMNeT++ Network Simulation Framework, http://www.omnetpp.org/