帯域見積りに基づく輻輳回避アルゴリズム

釘本健司[†,☆] 天海良治[†,☆]

TCPによるデータ転送において，TCPに輻輳回避のアルゴリズムが備えられているにもかかわらず，輻輳によりスループットが著しく低下する現象が発生する．この現象は，エンドノードでのネットワークの帯域よりも狭い帯域のリンクが転送経路上に存在するときに顕著である．本論文では，まずこの輻輳の発生過程を解析し，従来のTCPのアルゴリズムの問題点を明らかにする．次にスループットを向上させる輻輳回避アルゴリズムBECCを提案する．BECCでは転送経路上の最も狭い帯域を検出して，これに基づいて最大ウィンドウサイズを動的に制御することにより輻輳の発生を抑制し，スループットを改善する．BECCには既存のTCPプロトコルを変更する必要がないことと，アルゴリズムの変更は送信側に限られるという利点がある．実験用ネットワークにおける測定で，BECCが輻輳の発生を抑制し，スループットを向上させることができることを確認した．

Congestion Avoidance Algorithm Based on Available Bandwidth Estimation

TAKESHI KUGIMOTO[†,☆] and YOSHIJI AMAGAI[†,☆]

In TCP data transmission, there are cases when the built-in TCP congestion avoidance algorithm does not always work. In such cases, a significant number of packets are lost and re-transmitted, degrading the overall throughput. A congestion becomes most severe when there is a narrow band link along the routing path. This paper analyzes this phenomenon and proposes the BECC (Bandwidth Estimation-based Congestion Control algorithm) that suppresses these congestion. BECC algorithm has two advantages. One is that there is no need to change the TCP protocol, and the other is that only the sender is required to change its algorithm. Effectiveness of BECC is shown by practical comparisons against the existing algorithm.

1. はじめに

気象情報や遺伝子情報などを扱うビッグサイエンスの分野では，巨大データベースをもとに膨大な演算を行ってシミュレーションや解析を進める．近年では，より大規模で高度な演算を行うために，様々な組織の持つスーパコンピュータやデータベースを相互接続する必要が高まっており，これらの相互接続を可能とする広域高速インターネット技術が要求されている．

インターネットにおいて，TCP (Transmission Control Protocol)[1]による高いスループットで通信を行うためには，最大ウィンドウサイズを往復の遅延時間（RTT: Round Trip Time）と帯域の積よりも大きくする必要がある．この積はDBP（Delay Bandwidth Product）と呼ばれる[2]．しかし，広域高速インターネットでは，これまでのインターネットと様相が異なり，単にウィンドウサイズを大きくしただけでは帯域に見合ったスループットが得られない場合がある．

従来のTCPでは，輻輳によりパケット消失とその回復が定期的に繰り返され，データ転送が間欠的にしか行われない場合がある[3)~6)]．この間欠転送現象は，転送経路上にエンドノードよりも狭い帯域のリンク，すなわち帯域のボトルネックが存在するときに顕著に発生する．広域高速インターネットにおいては，この現象が発生するとスループットが特に著しく低下してしまう．その理由は，ネットワークの輻輳により消失したパケットの再転送や，輻輳状態を解消するまでの転送停止時間によるスループットへの影響が，低速ネットワークに比べて大きいからである．この間欠転送現象を招く輻輳の発生は，TCPの輻輳回避アルゴリズム[7]に帯域に見合った最大ウィンドウサイズを見積もる仕組みがないため，ウィンドウサイズを必要以上に拡大して過剰にデータを送信することに原因がある．

† 日本電信電話株式会社ソフトウェア研究所
NTT Software Laboratories
☆ 現在，日本電信電話株式会社光ネットワークシステム研究所
Presently with NTT Optical Network Systems Laboratories

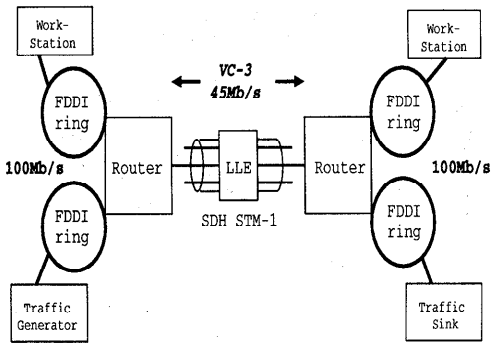


図1 測定環境 (1)
Fig. 1 Experimental network configuration (1).

本論文では，このTCPの間欠転送現象の発生を抑止してスループットを改善するための新しいアルゴリズムBECC（Bandwidth Estimation-based Congestion Control algorithm）を提案する．このアルゴリズムは，コネクションごとに使用可能な帯域を見積もる機能をTCPに与え，この見積りに基づいてウィンドウサイズの上限を動的に制御することにより輻輳を回避し，スループットの改善をはかるものである．また，このアルゴリズムの実装上の利点は，(1) 送り側のアルゴリズムのみの変更で済むこと，(2) TCPのプロトコルを変える必要がないことの2点である．

以降，2章ではTCPの間欠転送現象の発生過程とその原因について述べ，3章では，TCPの間欠転送現象を解決する輻輳回避アルゴリズムBECCについて述べる．4章では従来アルゴリズムとの比較による評価を行い，5章ではBECCの効果についてまとめる．

2. TCPの間欠転送現象

本章では，TCP RenoにTCP Extension[8)~10)]を追加したTCPを用いて，最大ウィンドウサイズを変化させたときのスループットの変化を調べる実験と，そこで観測された間欠転送現象およびその原因について述べる．

2.1 実験環境

スループットの測定は，**図1**のようなネットワーク環境を構築して実験を行った．

実験に使用した2台のワークステーションはFDDI（Fiber Distribution Data Interface）リングにつながっており，最大ウィンドウサイズを1034 bytesから1 Mbytesまで変化させることができる．TCPによるデータ転送は，この2台のワークステーションの間で行われる．また，動作中のTCPコード内部の変数の変化を観測できるようにカーネルに改造を施してある．

また，2台のルータの間はLLE（Long Link Emulator：遅延発生装置）を介してSDH STM-1（155 Mb/s）でつながっている．SDH STM-1は，仮想的に3本のVC-3（45 Mb/s）にわかれており，このうちの1本で結んである．このリンクでは，LLEにより100 msまでの伝送遅延を発生できる．

さらに，それぞれのルータのもう1つのFDDIリングにはTraffic GeneratorとTraffic Sinkが接続されており，この間に任意の速度でUDP（User Datagram Protocol）パケットを流すことにより，TCPコネクションが使用できる帯域を仮想的に変化させることができる．

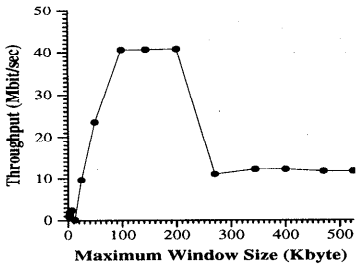


図2 最大ウィンドウサイズ vs スループット
Fig. 2 Maximum window size vs throughput.

2.2 最大ウィンドウサイズとスループット

図2は，最大ウィンドウサイズを1536 bytesから524 Kbytesまで変化させたときのスループットの変化である．転送データの総量は31 Mbytesであり，LLEで設定されているRTTは10 msである．UDPによるバックグラウンドトラフィックはない．

最大ウィンドウサイズが90 Kbytesまでの範囲では，スループットと最大ウィンドウサイズは比例しており，90 Kbytesから200 Kbytesの間はスループットは約41 Mb/sで飽和している．また，最大ウィンドウサイズが200 Kbytesを超えるとスループットの低下が起こり，約12 Mb/sまで落ちている．このように，最大ウィンドウサイズがDBPに対して大きすぎる場合には，スループットの著しい低下が起こることが分かる．このスループットの低下はTCPの間欠転送現象によるものである．以下では，この間欠転送現象について詳細に述べる．

2.3 間欠転送現象発生時のTCPの動作解析

TCPでは信頼性のある通信の実現のために送信側から送られたパケットに対して確認応答（ACK: Acknowledge）を行う．パケットの識別のために送信パケットにはSEQ（Sequence）番号が付与される．この番号はコネクションが確立したときに選ばれた任意の値に転送済みのデータ量をオフセットとして加算したものである．受信側はパケットを受信すると，確認

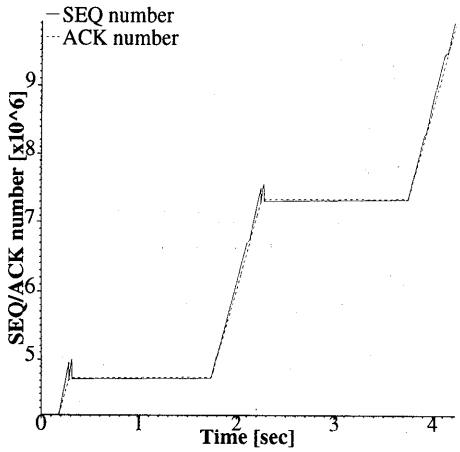


図3 間欠転送現象発生時の順序番号の増加
Fig. 3 SEQ number trace with a large maximum window size.

応答としてACKパケットを返す．ACKパケットにはACK番号が付与される．ACK番号は受信側が次に受けとることを期待しているSEQ番号である．

また，4.3BSD以降のTCPでは輻輳の回避のために，輻輳ウィンドウと呼ばれるウィンドウが用意されている．送信ウィンドウサイズが輻輳ウィンドウの値を超えないようにすることで，ネットワークの輻輳の状態に応じてパケットの送信を動的に制限し輻輳を回避する．

図3は，送信側で観測した，間欠転送現象発生中のSEQ番号とACK番号のそれぞれの増加の様子を表している．急激な転送と転送停止が交互に起きている．これがトータルのスループットを低下させる原因である．

図4は，図3のグラフ中の最初の輻輳の発生による転送停止前後の部分を拡大したものである．測定から得られたTCPの内部変数の値を解析した結果とあわせ，間欠転送現象が起こっているときのTCPの動作をこの図を用いて説明する．

(1) 送信側の輻輳ウィンドウの値がACKにより増加し，200 Kbytesを超える．一方，最適な最大ウィンドウサイズとは図2においてスループットが飽和し始める点であり，約90 Kbytesである．輻輳ウィンドウの値はこれに比べて大きいので，使用可能帯域以上にパケットをネットワークに送り込んでしまうために輻輳が起こる．このため送信側のパケットの連続したSEQ番号の複数のパケットが消失する（図4の白マル）．

(2) 送信側パケットの消失のため，受信側は期待しているものよりも大きなSEQ番号を持つパケットを受信する．これらを受けとるたびに受信側は未受信のSEQ番号をセットしてただちに送信側にACKを返す（図4の+）．このときのACKパケットのACK番号は重複しているので，重複ACK（duplicate ACK: dup-ACK）と呼ばれる．

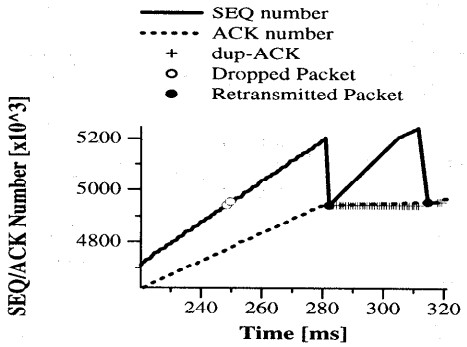


図4 輻輳発生時のSEQ番号
Fig. 4 SEQ number trace during congestion.

(3) 消失したパケットが送り出された直後の送信側のウィンドウは十分大きいため，送信側がdup-ACKにより輻輳を検出する前に，先行して多くのパケットが送られる（図4の250～280 ms）．これらのパケットの数だけdup-ACKが返されることになる．

(4) 送信側は，受信側から送られてきたdup-ACKの最初の3個を受けとると，消失したパケットを回復するためにFast Retransmitアルゴリズム[11),12)]によって当該パケットだけの再送を行う（図4の黒丸）．その直後にFast Recoveryアルゴリズムにより輻輳ウィンドウの値は半分にされるため，送信側はパケットの送信を再開しようとするが，送達が確認されていないデータの量に比べて輻輳ウィンドウは小さいため，パケットの送信ができない．

(5) 一方，受信側からはすでに送信されたパケットに対するdup-ACKが返ってくる．dup-ACKはネットワークの潜在的な帯域と見なされるため，送信側ではACKが返るたびに輻輳ウィンドウがMSS（Maximum Segment Size）ずつ増やされる．こうして増加した輻輳ウィンドウの値が，送達が確認されていないデータの量を超えるとパケットの送信が再開され，いくつかのパケットが送られる（図4の305～315 ms）．Fast Retransmitで再送したパケットに対するACKが返ると，大きくなり過ぎた輻輳ウィン

ドウの値を是正するために，輻輳ウィンドウはFast Recovery直後の値にまで減じられる．

(6) 送信側から再送したパケットの次のパケットも消失しているので，再びこのパケットに対するdup-ACKが続く．ここで再びFast Retransmit/Fast Recoveryアルゴリズムが働く．しかし，先行して送られたパケットが1度目に比べると少ないためdup-ACKも少なく，パケットが送信可能となるほど輻輳ウィンドウが十分に大きくならない．こうして送信側からのパケットの送信がまったく行われなくなり，TCPタイマアルゴリズムにより送信が再開されるまでの間，1～1.5秒間転送は完全に止まる．この一連の動作が定期的に繰り返されるためにスループットが低下する．

以上の解析から，従来TCPの間欠転送現象の原因は，TCPが使用可能帯域を見積もって送信を適切に制御する機能を持たないため，使用可能帯域を超えてパケットを送信しようとして輻輳を定期的に起こしてしまうことである．

現在，この現象を解決する輻輳制御アルゴリズムとしては，FACK[13]（Forward Acknowlegment）が提案されている．FACKはSACK[14),15)]（Selective ACK）をベースとした輻輳制御アルゴリズムであり，輻輳からの回復を主眼としたものである．SACKは，受信側に正常に届いたデータブロックを送信側に知らせて失われたパケットだけを再送させ，データの不必要な再送を防いでスループットを向上させることを目的としたTCPの拡張である．FACKは，このSACKに間欠転送現象を防ぐ機能を追加したものである．FACKでは，ネットワーク中に滞留しているデータ量によって，TCPの再送タイムアウトが起こらなくてもパケットの送信を継続する．

これに対して，我々が開発したアルゴリズムは輻輳の回避を目的としている．次章では，このアルゴリズムについて述べる．

3. BECCアルゴリズム

本章では，従来のTCPの間欠転送現象を抑制し，安定かつ高速な通信を行う目的で我々が新たに開発したBECC（Bandwidth Estimation-based Congestion Control algorithm）について述べる．開発にあたっては，BECCを組み込んでいないホストとの通信でも効果を得るために，TCPのコードの改造は送り側の改造だけで済むようにすることと，TCPプロトコル自体には変更を加えないことを方針としている．

以下ではアルゴリズムの概要について述べ，次にこのアルゴリズムで用いる使用可能帯域の見積り手法と，正確なRTTの計測法について述べたのち，このアルゴリズムの動作の詳細について述べる．

3.1 アルゴリズムの概要

間欠転送現象はパケットの送り過ぎによる輻輳の発生に起因し，パケットの送り過ぎは，ウィンドウサイズが使用可能帯域に対して適切な値を超えることにより起こる．したがって，この間欠転送現象を回避するには，(1) 使用可能帯域を見積もり，(2) その見積りに基づいて送信を制御し，輻輳を抑制すればよい．

BECCでは使用可能帯域を超えないようにパケットの送信を制限する目的で，帯域ウィンドウと呼ぶ新しいウィンドウを導入する．送信のウィンドウサイズは，この帯域ウィンドウの値と，受信側から通知されたウィンドウサイズ，輻輳ウィンドウの値のうち最も小さいものとする．帯域ウィンドウの値には，その時点で見積もった使用可能帯域とRTTの最小値の積（DBP）の値を使用する．使用可能帯域として実転送レートの最大値を用いる．また，RTTの最小値を用いる理由は，輻輳の発生時のRTTは伝送遅延でだけでなく，負荷の増大による処理時間を含んでしまうため，正しいDBPの値が得られないからである．

帯域ウィンドウの更新は，輻輳の発生時および1.5秒ごとに呼び出されるタイマ関数により行われる．輻輳の発生を契機として行われる更新は，送信制限により輻輳を抑止することが目的である．輻輳が発生すると，使用可能帯域と最小RTTの積が帯域ウィンドウの値としてセットされる．これにより，使用可能帯域に見合った速度でパケットが送り出される．前章の図2のスループットの測定において，スループットが飽和しているときのTCPの内部変数の値を調べると，RTTは16 msであり，実転送レートの最大値（使用可能帯域）は約41 Mb/sである．したがって，その積は82 Kbytesである．一方，最適な最大ウィンドウサイズはスループットが飽和し始める点であるので，図2から約90 Kbytesと読みとれる．このことは，使用可能帯域と最小RTTの積の値を帯域ウィンドウの値として使用することで，使用可能帯域に見合った速度でパケットを送り出せることを裏付けている．

また，タイマ関数により行われる更新は，使用可能帯域の拡大を検出することが目的である．タイマ関数が呼び出されると，帯域ウィンドウの値を拡大し，実転送レートが増えるかどうかを見ることで使用可能帯域の拡大を検出する．実転送レートとは受信側が実際に受けとれた単位時間あたりのデータ量である．実転

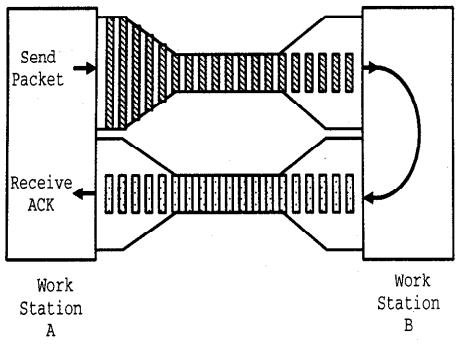


図5 使用可能帯域の見積り
Fig. 5 Estimation of available bandwidth.

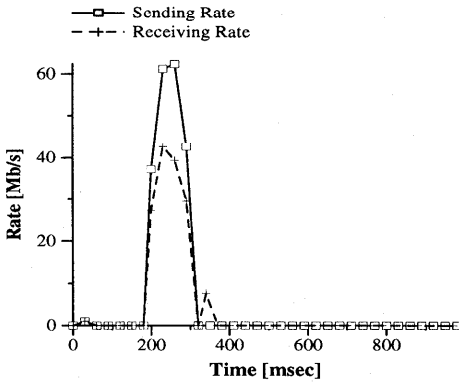


図6 送信レートと実転送レート
Fig. 6 Sending rate and the actual transmission rate.

送レートが増えていなければ使用可能帯域に変化がないとして帯域ウィンドウの値を元に戻し，増えていれば余裕があると見て帯域ウィンドウをさらに拡大する．

以下では，ネットワークの使用可能帯域の見積り手法と，正確なRTTの計測法について述べ，次にBECCの機能として，帯域ウィンドウによる送信制限，帯域ウィンドウ更新について述べる．

3.2 ネットワークの使用可能帯域の見積り手法

これまでTCPにはネットワークの使用可能帯域を見積もる機能はなかった．ここでは，送信側に返るACK番号の増加率からその時点での使用可能帯域を見積もる手法を提案する．

図5に示すように，受信側のデータの受信速度は，転送経路上の最も低速なリンクの帯域に一致する．受信側では，到着したパケットに対してただちにACKパケットが送信側に送り返される．したがって，送信側に返ったACKパケットのACK番号の増分から算出される単位時間あたりの転送データ量は，ネットワークを通して受信側に到着した実転送レートに等しい．この実転送レートの最大値が，このコネクションにおける使用可能帯域である．また，ACKパケットのサイズは十分に小さいため，通常はACKの転送には転送経路上の帯域のボトルネックはないと考えてよい．ACKが遅れることで送信方向の使用可能帯域を実際よりも小さく見積もってしまう可能性があるが，ACKが遅れることは応答方向のネットワークの使用可能帯域が小さいことを示しているので，これが実転送レートとなる．

図6は，従来TCPの転送の様子を示した図3の0～800 msの間の送信側でのSEQ番号の増加率（送信レート）と，返ってきたACK番号の増加率（実転送レート）をグラフにしたものである．実転送レートはこの値以内に収まっており，その最大値が使用可能帯域である．

ACKの増加により使用可能帯域を見積もる機能をTCPに追加するための準備として，関数`swnd_timer()`を用意し，変数`t_acked`, `t_maxacked`, `ms_period`, `t_msrd_th`をTCPコネクションごとに導入する．`swnd_timer()`は，周期`ms_period`で呼び出されるタイマ関数であり，`t_acked`は，1周期の間に受信側への送達が確認されたデータ量を格納する変数である．変数`t_maxacked`には`t_acked`の最大値が格納される．変数`t_msrd_th`には，`t_acked`のこれまでの変動を吸収した現在の実転送レートが以下の式から計算されて記録される．

$$\texttt{t\_mfsrd\_th} = \alpha \times \texttt{t\_msrd\_th} + (1-\alpha) \times \texttt{t\_acked/ms\_period} \quad (1)$$

αは`t_msrd_th`の変動と収束をコントロールする係数で，大きいほど収束が速くなるが変動は大きくなる．BECCではシステムへの負荷を小さくするためにαの値を0.75としてビットシフトによる処理を行っている．

`t_acked`と`t_maxacked`の単位はbyteであり，`t_msrd_th`の単位はbyte/secである．

これらの関数と変数を使って，ネットワークの使用可能帯域を以下のようにして見積もる．

(1) ACKパケットが受信されると，受信パケットを処理するTCPの関数である`tcp_input()`が起動される．`tcp_input()`では，ACK番号の増加分だけ`t_acked`を増加させる．

(2) `tcp_input()`とは独立に`ms_period`ごとにタイマ関数`swnd_timer()`が呼び出される．この時点で，`t_acked`は，最近の`ms_period`の間に送達が確認されたデータの量を表す．`t_acked`と`t_maxacked`の大きい方を`t_maxacked`と

する．t_maxacked を周期 ms_period で割ったものが使用可能帯域である．また，実転送レート t_msrd_th は t_acked の値から式(1)で計算される．この後，t_acked を次の周期の送達確認データ量を記録するために0でクリアする．

関数 swnd_timer() の呼び出し周期 ms_period は，短かすぎても長すぎても正確なものが得られない．本アルゴリズムでは，呼び出し周期 ms_period を経験的に最も良い値と思われる30 msごととした．

3.3 RTTの正確な計測法

BECC では TCP Extension の Time Stamp Option を利用して1 msの精度で RTT の最小値の計測ができるように TCP コードの改造を行った．RTT の最小値の記録のために t_minrtt を新しく用意した．

Time Stamp Option を用いて RTT は以下のように計測される．まず，パケットの送信にあたって TCP は4 byteのオプションフィールドにシステム起動時からの時刻を表すシステム内部変数 tcp_now の値を記録する．受信側は，受けとったパケットのオプションフィールドの値を ACK パケットのオプションフィールドに入れて送り返す．送信側は ACK のオプションフィールドに記録された時刻と tcp_now の差分をとる．この差分が RTT であり，変数 t_rtt に記録される．BECC では tcp_now に1 ms単位のシステム時刻を記録する．t_rtt と t_minrtt のうち小さい方を t_minrtt に代入し，RTT の最小値を記録する．

3.4 帯域ウィンドウの決定と更新および送信制限

ここでは BECC の機能である帯域ウィンドウによる送信制限と，帯域ウィンドウの更新について述べる．

輻輳を契機とした帯域ウィンドウの決定と更新

BECC を組み込んだ TCP では，コネクションが確立した直後は従来のアルゴリズムでの動作とまったく同じように通信を開始する．初期状態では BECC は送信制限をしないためである．その理由は，使用可能帯域の見積りは ACK に基づくため，使用可能帯域を正しく見積もるには，パケットをネットワーク中にある程度は送り込む必要があるためである．

輻輳が起こってパケットが消失すると，Fast retransmission / FastRecovery アルゴリズムが働き始める．これを契機として帯域ウィンドウを設定する．具体的には，4つの dup-ACK が返ったときに BECC は帯域ウィンドウの値を設定する．帯域ウィンドウの値は，最大の実転送レート（使用可能帯域）と RTT の最小値 t_minrtt の積である．ここで使用可能帯域として t_maxacked を swnd_timer() の呼び出し周期 ms_period で割ったものを用いる．また，RTT の最小値を用いる理由は，輻輳の発生によりルータの負荷も増加するため，帯域ウィンドウの値を決定するアルゴリズムが動作するときには RTT も増加してしまい，DBP の計算上適切な値ではなくなるためである．最後に t_maxacked の値を，次回の使用可能帯域の見積りのために0にクリアする．

帯域ウィンドウによる送信制限

決定された帯域ウィンドウ snd_swnd を TCP パケットを送信する関数 tcp_output() の中のウィンドウを決定するコードに以下のように組み込むことより，高速ネットワークの帯域と遅延にあった適正な量のパケットが送信される．

```
win = min3(snd_wnd, snd_cwnd, snd_swnd);
```

snd_wnd と snd_cwnd は，それぞれ受信側から通知されてきたウィンドウ，輻輳ウィンドウである．受信側から通知されてきたウィンドウの値は受信側が現在受けとることのできる最大のデータ量を表しており，輻輳ウィンドウの値は現在の輻輳の発生状態に応じて，輻輳が悪化しないようにパケット送信を動的に制限するためのウィンドウサイズの上限を表している．本アルゴリズムで追加した帯域ウィンドウは，ネットワークの使用可能帯域を超えてパケットを送信しないようにウィンドウサイズの上限を決めるものであり，輻輳ウィンドウほど動的には変化せず，半固定的なものである．これらの3つのウィンドウのどれを超えても輻輳や再送が発生してしまうので，送信側のウィンドウサイズはこれら3つのウィンドウのうちの最も小さいものをとる．

定期的な帯域ウィンドウの更新

ネットワークの使用可能帯域は時間とともに変化するので，いったん帯域ウィンドウが設定されると，過去に見積もられた帯域に合わせてパケットの送信が行われ続ける．そのため他のトラフィックの減少等により使用可能帯域が広がっても，測定される実転送レートは ACK に基づいているため変化せず，使用可能帯域の拡大を検出できない．そこで，定期的に帯域ウィンドウの値を拡大し，実転送レートが増えるかどうかを見ることで使用可能帯域の拡大を検出する．実転送レートが増えていなければ使用可能帯域に変化がないとして帯域ウィンドウの値を元に戻し，増えていれば余裕があると見て帯域ウィンドウをさらに拡大する．

図2のグラフにより，最適なウィンドウサイズ（スループットが飽和し始める点）の2倍程度を超えない範囲ならば，使用可能帯域の変化を探るためにウィンドウサイズを変化させることができる．しかし，BECC が数多くのホストで実装された場合，帯域ウィンドウ

のサイズの増加量が大きすぎるとネットワークにかける負荷が高くなり輻輳を起こす可能性がある．そこで，ウィンドウの増加量はできるだけ小さくする必要がある．実験の結果，1.2倍よりも小さな倍率にすると，スループットの変化を検出することが難しくなることが分かった．また，多くのUNIXのカーネル内部では浮動小数の演算は禁止されているため，小数の乗算をビットシフトと加算を組み合わせて $x = x + (x \gg 2)$ のように行った．

BECCでは，定期的な更新の動作を行うために，関数 `swnd_update()` を追加した．この関数の動作モードには，`UPDATE_START`, `UPDATING`, `NOT_UPDATE` の3つのモードがある．`UPDATE_START` は更新を開始し，帯域ウィンドウを拡大してもよいモードである．`UPDATING` はネットワークの使用可能帯域の拡大を検出するために，帯域ウィンドウの値を変化させたことを示すモードである．また，`NOT_UPDATE` は帯域ウィンドウの更新をしてはならないことを示すモードである．帯域ウィンドウの更新のアルゴリズムは，`UPDATE_START` のモードから開始される．

関数 `swnd_update()` は1.5秒ごとにタイマで呼ばれ，それぞれのモードに応じて以下の動作を行う．

(1) `UPDATE_START`

ここでは，変数 `expct_th` と `pre_snd_swnd` を導入する．変数 `expct_th` は `snd_swnd` を増加させたことにより期待できる実転送レートの評価に使用する．変数 `pre_snd_swnd` は拡大される前の `snd_swnd` の値を記録しておくために使用する．

使用可能帯域の拡大を検出するために，現在の帯域ウィンドウ `snd_swnd` の内容を `pre_snd_swnd` に記録し，`snd_swnd` の値を1.25倍する．1.25倍の演算は，ビットシフトと加算で実現できる演算のうち，1.2倍に最も近いものである．次に現在の実転送レートを表す `t_msrd_th` の値をビットシフトと加算を組み合わせて1.125倍したものを `expct_th` に記録し，更新のモードを `UPDATING` に変える．理論的には実転送レートは帯域ウィンドウの増加に比例して増加するはずだが，実際には必ずしも比例していない．そこで，帯域ウィンドウの増加は1.25倍とし，期待されるスループットは元の値の1.125倍としている．

(2) `UPDATING`

`swnd_update()` が `UPDATE_START` のモードで呼ばれたときに設定された，期待される実転送レート `expct_th` の値と現在の実転送レートである `t_msrd_th` の値を比較する．

`t_msrd_th` の方が大きければ，使用可能帯域に余裕があると見て `snd_swnd` の値を `pre_snd_swnd` に記録し，さらに `snd_swnd` を1.25倍に拡大する．また，期待される実転送レート `t_expct_th` を1.125倍する．このとき，更新のモードは `UPDATING` に保つ．

逆に `t_msrd_th` が小さければ，TCPがすでに使用可能帯域を使い切っていると判断して，`snd_swnd` の値を `pre_snd_swnd` に戻し，更新のモードは `UPDATE_START` とする．

(3) `NOT_UPDATE`

このモードは，輻輳によりdup-ACKが4つ返ってきたときか，`snd_swnd` の方が `snd_cwnd` よりも大きいときに，それぞれ `tcp_input()` と `tcp_output()` の中で設定される．

輻輳が起こると，輻輳を契機とした帯域ウィンドウの値の決定が行われるため，更新処理は不要となる．また，帯域ウィンドウ `snd_swnd` が輻輳ウィンドウ `snd_cwnd` よりも大きいときには，ウィンドウサイズは輻輳ウィンドウによって制限されているため，帯域ウィンドウの更新は無意味である．そこで，モードを `UPDATE_START` に変えるだけで更新に関する動作を行わない．

4. 評　価

本章では，まず従来のTCP Reno + TCP Extension（RE-TCP）と，BECCアルゴリズムを実装したTCP（BECC-TCP）をホストに実装して行った測定の結果の比較を通じて，BECCアルゴリズムの効果について述べる．次にBECCアルゴリズムの実装上のコストを評価する．

4.1 BECCの効果

図7 の実験環境で，BECC-TCPとRE-TCPでの測定を行った．ボトルネックとなるリンクの帯域やメディアの種類にかかわらず本アルゴリズムが動作することを示すために，図7の(1)，(2)をつなぎ替えることにより，VC-3リンク（45 Mb/s）とEthernet（10 Mb/s）の2つにより測定を行えるようにした．Traffic GeneratorからTraffic Sinkへ任意の速度でUDPパケットを流すことにより，TCPコネクションが使用できる帯域を仮想的に変化させる．また，この測定での総データ転送量は114 Mbytesであり，またソケットバッファサイズの調整により，TCPのとりうる最大ウィンドウサイズは1048576 bytesとしてある．

4.1.1 TCPの振舞い

図8 と **図9** は，RE-TCPとBECC-TCPによる転送でのSEQ番号の増加を表したものである．図8は，

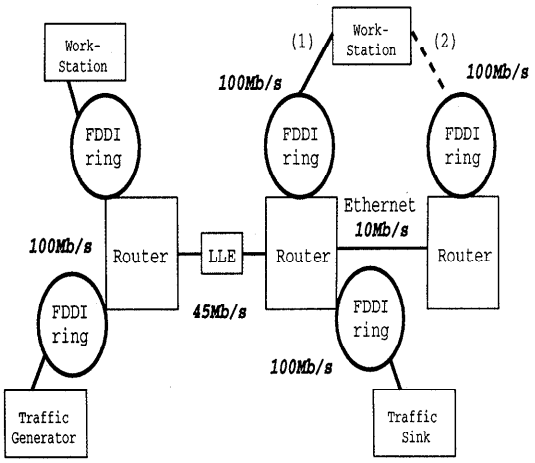


図7 測定環境 (2)
Fig. 7 Experimental network configuration (2).

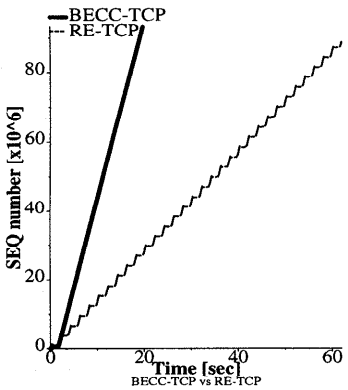


図8 SEQ 番号の増加 (1)
Fig. 8 SEQ number trace (1).

図7中の (1) のように FDDI リングに接続したときのものであり，図9は，(2) のように接続したものである．

BECC-TCP，RE-TCP ともにコネクションが確立してから 0.5 秒以内に輻輳を起こし，再送タイムアウトとなるまで転送が停止している．RE-TCP では，この後も間欠転送現象が起こるのに対し，BECC-TCP では順調に転送が行われる．

図7の (1) のように接続して測定した際の BECC-TCP の内部から取得された変数の値によると，Fast Retransmit/Fast Recovery が最初に働く直前の `t_maxacked` は 270900 bytes であり，最小の RTT `t_rttmin` は 14 ms である．また，再送タイムアウト後に設定される帯域ウィンドウの値は 126420 bytes である．帯域ウィンドウによってパケットの送信がネットワークの使用可能な帯域に応じて適切に制限されるため，輻輳を起こすことなく転送が行われる結果，RE-TCP よりも高いスループットが得られることが分かる．

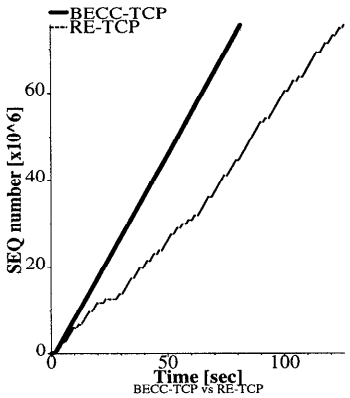


図9 SEQ 番号の増加 (2)
Fig. 9 SEQ number trace (2).

4.1.2 スループット

図10 と**図11** は，RE-TCP と BECC-TCP の両方で 30 Kbytes から 350 Kbytes まで最大ウィンドウサイズを変えてスループットを測定したものである．図10は図7の (1) のように，図11は図7の (2) のようにそれぞれ接続した場合のものである．各々の測定点では，4000 bytes のパケットを 2000 回，総計 31.25 Mbytes のデータ転送を行っている．

図10では，どちらの TCP も最大ウィンドウサイズが 220 Kbytes 以下の領域では，ほぼ同様のスループット特性が得られる．間欠転送現象の発生し始める 220 Kbytes 以降の領域では，RE-TCP が約 12 Mb/s のスループットしか得られないのに対して，BECC-TCP では平均して約 30 Mb/s のスループットが得られる．同様に，図11でも，間欠転送現象の起こる領域においては RE-TCP よりも BECC-TCP の方が高いスループットが得られることが分かる．しかし，図10と比べると，スループットの差は小さい．この図からも低速ネットワークにおいては間欠転送現象の影響が小さいことが分かる．このようにネットワークが高速になるほど，間欠転送現象によるスループットの低下が著しい．

すでに述べたように，BECC-TCP では基準点となるネットワークの帯域を見積もるために，はじめの1回だけは RE-TCP と同様に輻輳が起こるまでパケットを送り込む．そのため，この領域では転送の停止が1度は起こる．したがって，この領域ではスループットはネットワークの使用可能帯域までには到達できないが，1回の TCP コネクションでのデータ転送量が

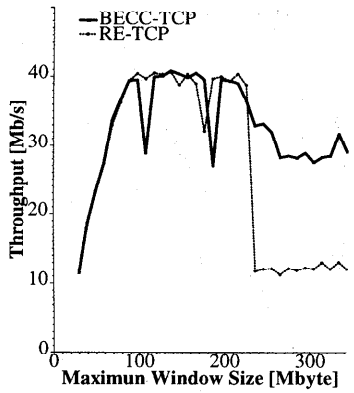


図10 BECC-TCP vs RE-TCP (1)
Fig. 10 BECC-TCP vs RE-TCP (1).

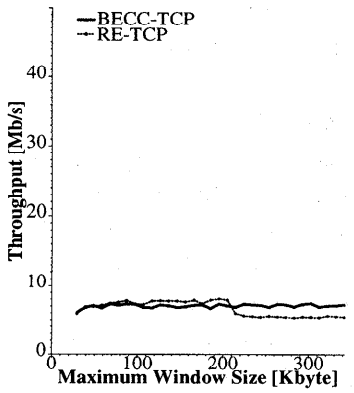


図11 BECC-TCP vs RE-TCP (2)
Fig. 11 BECC-TCP vs RE-TCP (2).

多くなるにつれ，この転送停止による影響は小さくなるため，スループットはネットワークの使用可能帯域に近付く．

4.1.3 帯域ウィンドウ更新アルゴリズムの効果

帯域ウィンドウ更新アルゴリズムの効果を調べるため，転送開始から10秒後までと20秒後から転送終了までの間に，Traffic Generatorを使用して，約20 Mb/sのUDPによるバックグラウンドトラフィックを45 Mb/sリンクに流した．

図12はBECC-TCPのSEQ番号の増加の様子と，内部で測定された現在の実転送レート`t_msrd_th`の変化を示したグラフである．転送開始後から10秒後までの間は，初めに輻輳が起こって転送が止まるが，帯域ウィンドウにより送信が適切に制限されてパケットが順調に送信されている．このときの実転送レートは約15 Mb/sである．10秒後から20秒後の間はUDPによるバックグラウンドトラフィックがなくなるため，TCPの使用可能な帯域が広がる．BECC-TCPは1.5

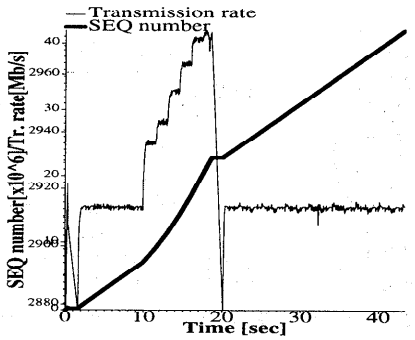


図12 SEQ番号と実転送レートの変化
Fig. 12 SEQ number trace and the actual transmission rate.

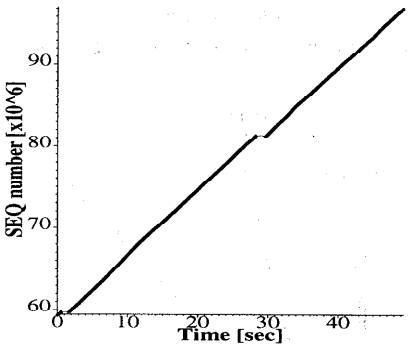


図13 2ストリームの転送におけるSEQ番号の増加
Fig. 13 SEQ number trace with two BECC-TCP stream.

秒ごとに使用可能な帯域を探っているため，これにともなって`snd_swnd`を増加させ続ける．`t_msrd_th`が上昇していることからBECC-TCPの帯域更新のアルゴリズムが有効に働いていることが分かる．転送開始20秒後に再びUDPによるバックグラウンドトラフィックが流れて使用可能帯域が減少すると，輻輳により転送がいったん停止するが，BECC-TCPは再び使用可能帯域を見積もり直して帯域ウィンドウによる制限が効き始めるため，その後の転送はスムーズに行われる．

4.1.4 2ストリームのBECC-TCPの相互干渉

図13は，ワークステーション間に2つのBECC-TCPストリームを同時に流したときの片方のストリームのSEQ番号の増加を示したものである．どちらのストリームも帯域を適切に測りながらデータの転送を行うため，途中若干の輻輳は起こるものの，安定して転送が行われている．

4.2 BECC-TCP実装のコスト

本節では，BECC-TCPの実装上のコストを，使用メモリ量の増加量，TCPの入出力処理とタイマ処理

のシステム負荷の増加量の3つの観点から評価する．

4.2.1 メモリ

BECC-TCPではTCPコントロールブロックと呼ばれる構造体中の変数をいくつか増やしている．内訳はunsigned longを6つ，intを4つそれぞれ追加している．実験に用いたワークステーションでは，unsigned long，intともに4バイトであるため，これらの総計は40バイトである．TCPコントロールブロックはコネクションごとに作成されるので，BECCを使うことにより，Nストリームにおけるメモリ使用量が$40 \times N$ [bytes]増える．現在のワークステーションに一般的に使用されているメインメモリ空間が64 Mbytes程度であることを考えると，この値は無視することができる．

4.2.2 入出力処理

BECCでは，`tcp_input()`と`tcp_output()`に輻輳の検出と帯域の見積りのための新しいコードが追加されている．`tcp_input()`では，(1) 正確なRTTの見積りのために，マイクロ秒の時刻の取得のための関数呼び出し，2つの除算，2つの乗算，1つの条件分岐，1つの代入が追加され，(2) 輻輳の検出のために，1つの乗算と1つの除算，2つの条件分岐と3つの代入が追加されている．`tcp_output()`では，(3) ウィンドウの値の決定のために，条件分岐が2つと代入が1つ追加され，(4) RTTの正確な見積りのために，マイクロ秒の時刻を取得する関数呼び出し，2つの除算，2つの乗算，1つの条件分岐，1つの代入が追加されている．

これらの実行時間はTCPの入出力の処理全体からみればきわめて短いものであり，新しいコードの追加による影響を無視できる．

4.2.3 タイマによる処理

BECC-TCPでは，実転送レート測定のために30 msごとに呼び出されるタイマ関数`swnd_timer()`と，帯域ウィンドウを更新するために1.5秒ごとに呼ばれるタイマ関数`swnd_update()`が追加されている．`swnd_timer()`で主に行われる処理は，6つの代入，2つの条件分岐，1つの乗算，1つの減算，2つの除算である．これらは，TCPコネクションごとに実行される．

5. おわりに

本論文では，従来のTCPの輻輳回避アルゴリズムが，ウィンドウサイズを最大値まで拡大して帯域をできる限り使おうとして輻輳を起こすため，間欠転送現象が起こすことを示し，特に広域高速インターネットにおいては，この現象が著しくスループットを低下させることを示した．この間欠転送現象を回避するために，ACKの受信レートに基づく輻輳回避アルゴリズムであるBECCを提案した．BECCの実装上の利点は以下の2つである．

(1) TCPプロトコルを変更する必要がない
(2) アルゴリズムの変更は送信側だけで済む

また，BECCを実装したTCPを用いた実験を通して，以下の2つの性能上の効果を確認した．

(1) 使用可能帯域を実測しながら適切に送信を制限するため輻輳が頻発せず，従来TCPよりも高いスループットが得られる
(2) 他のトラフィックの増加による輻輳にも，他のトラフィックの減少による使用可能帯域の拡大にも追従する

今後は，BECC-TCPを実際のネットワークに適用し，アルゴリズムの改良を行うとともに，有効性の確認を進めていく予定である．

謝辞 本研究に関して議論していただいた，NTT光ネットワークシステム研究所村上健一郎グループリーダに感謝いたします．

参考文献

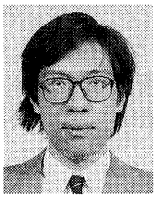
1) Postel, J.: Transmission Control Protocol, RFC793 (1981).
2) Kleinrock, L.: The Latency/Bandwidth Tradeoff in Gigabit Networks, *IEEE Communication Magazine*, Vol.30, No.4, pp.36–40 (1992).
3) Kugimoto, T., Murakami, K., Amagai, Y. and Oka, A.: An experience with a Long Fat Pipe, *9th ICOIN* (1994).
4) 天海良治，村上健一郎，釘本健司，岡 敦子，伊藤正樹，後藤滋樹，伊藤光恭：長距離超高速インターネットの特性解析，マルチメディア通信と分散処理研究会 (1994).
5) Amagai, Y., Murakami, K., Kugimoto, T. and Oka, A.: An Analysis of Behavior in a High-speed Long-distance Internet, *2nd JAIN Consortium Symposium* (1994).
6) 釘本健司，天海良治，村上健一郎：絶対帯域見積もりに基づく輻輳回避アルゴリズム，マルチメディア通信と分散処理研究会十和田ワークショップ (1996).
7) Jacobson, V.: Congestion Avoidance and Control, *SIGCOMM'88*, pp.314–329 (1988).
8) Jacobson, V. and Braden, R.: TCP Extensions for Long-Delay Paths, RFC1072 (1988).
9) Jacobson, V., Braden, R. and Zhang, L.: TCP Extension for High-Speed Paths, RFC1185 (1990).

10) Jacobson, V., Braden, R. and Borman, D.: TCP Extensions for High Performance, RFC1323 (1992).
11) Jacobson, V.: modified TCP congestion avoidance algorithm, end2end-interest mailing-list (1990).
12) Stevens, W.: TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms, RFC2001 (1997).
13) Mathis, M. and Mahdavi, J.: Forward Acknowledgment: Refining TCP Congestion Control, *ACM SIGCOMM*, Vol.26, No.4 (1996).
14) Mathis, M., Mahdavi, J., Floyd, S. and Romanow, A.: TCP Selective Acknowledgment Options, RFC2018 (1996).
15) Fall, K. and Floyd, S.: Simulation-based Comparisons of Tahoe, Reno, and SACK TCP, *Computer Communications Review* (July, 1996).

(平成9年 5 月12日受付)
(平成9年10月 1 日採録)

釘本 健司（正会員）

平成2年横浜国立大学工学部電子情報工学科卒業．平成4年同大学大学院工学研究科電子情報工学専攻博士課程前期修了．同年日本電信電話（株）入社．以来，計算機ネットワーク，知識処理の研究に従事．現在光ネットワークシステム研究所研究主任．電子情報通信学会，ACM各会員．



天海 良治（正会員）

昭和34年生．昭和58年電気通信大学電気通信学部計算機科学科卒業．昭和60年同大学大学院修士課程修了．同年日本電信電話（株）入社．以来，プログラミングパラダイム，計算機アーキテクチャ，計算機ネットワークの研究に従事．現在光ネットワークシステム研究所主任研究員．ソフトウェア科学会会員．