クラウドコンピューティング向けネットワーク高速化技術

Network Acceleration Technology for Cloud Computing

矢崎 武己 Yazaki Takeki
磯部 隆史 Isobe Takashi

日立グループは，クラウドコンピューティングにおいて必須な広帯域通信の実現に向け，高速TCPの研究開発を推進している。現在のネットワークで広く使用されるTCPは，広域な通信環境での通信遅延やパケット廃棄の影響により，通信帯域が大きく低下する課題があった。開発した高速TCPは，トークンバケットアルゴリズムによる通信遅延に非依存な帯域制御，パケット廃棄率の変化に基づく輻輳制御，廃棄パケットの迅速な再送信を行う再送制御により，通信帯域を向上する。評価実験の結果，日米間の通信を想定した環境で，TCPの23.6倍の通信帯域を達成した。今後は，この技術のクラウド適用に加え，適用時の運用管理を効率化する技術の研究開発を推進する。

1. はじめに

クラウドコンピューティングは，導入の容易性，低コスト性から企業での利用が急速に普及し，IT（Information Technology）システムの大きな潮流となっている。クラウド環境では，ユーザー拠点のITシステムが提供してきたサービスをネットワーク経由でデータセンターが提供するため，従来サービスと同様の信頼性や応答性などのサービス品質が求められる。そのため，ユーザーとデータセンターをつなぐネットワークには，信頼性に加え，通信の広帯域性が必要となる。

現在のネットワークでは，通信規約としてTCP/IP（Transmission Control Protocol / Internet Protocol）[1)~3)]が一般的に利用されている。1974年に開発されたTCPは，現在の広帯域な広域網を想定しておらず，通信距離の増加に伴い，通信遅延時間が増加し，通信帯域が低下する課題があった。この低下は，ユーザーに対するサービスの応答性の劣化を招き，ひいては企業の生産性を著しく低下させてしまう。クラウドコンピューティングは企業の基幹業務やわれわれの社会生活を支える社会インフラとなることが期待されており，ネットワークのサービス品質向上は，避けて通れない課題であった。

そこで，日立グループは，この課題の解決に向けた独自の高速TCPの研究開発を行っている。長距離通信における通信帯域を大きく向上することで，社会インフラをはじめとする多様な領域へのクラウドコンピューティングの普及が期待できる。

ここでは，クラウドコンピューティング向けネットワークの高速化技術について述べる。

2. TCPとその課題

TCPによる通信の概要とその課題について説明する。TCPでは，通信開始前に送受信端末間であらかじめコネクションを設定し，送信端末がパケットの送信を開始する。受信端末は受信したパケットに対応したACK（Acknowledgement）を送信端末に送信し（図1参照），送

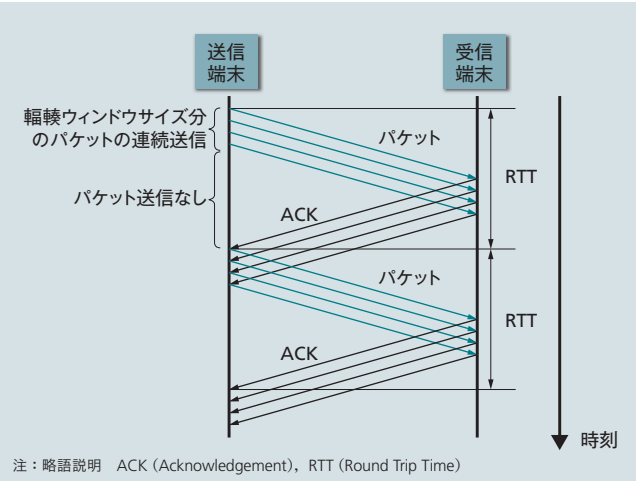


図1 | 輻輳ウィンドウサイズを用いた帯域制御

TCPでは，送信端末は輻輳（ふくそう）ウィンドウサイズ分のパケットを連続送信する。対応するACKの受信後に送信を再開するため，RTTの増加により通信帯域が低下する。

信端末はACKの到着によって送信パケットの到着／廃棄を認識する。廃棄されたパケットについては，再度送信することでデータ欠落のない通信を実現する。この再送信の仕組みを再送制御と呼ぶ。また，送信端末はネットワークの輻輳（ふくそう）状況を判断し，通信帯域を制限してパケットを送信する。この制御を輻輳制御と，特定の通信帯域に制限して送信する仕組みを帯域制御と呼ぶ。TCPの通信帯域の低下は，これらの3制御に起因している。

2.1　帯域制御

帯域制御は，ネットワークの輻輳を抑止するために，通信帯域を制限する送信端末の制御である。送信端末は，ACKの受信なしに送信するデータ量である輻輳ウィンドウサイズを動的に調整することで通信帯域を制御する[3), 4)]。輻輳ウィンドウサイズの最大値は，コネクション設定時に受信端末から通知されるパケット受信用の受信バッファ量（byte）と，送信端末の送信用のバッファ量の小さい値から決まる。

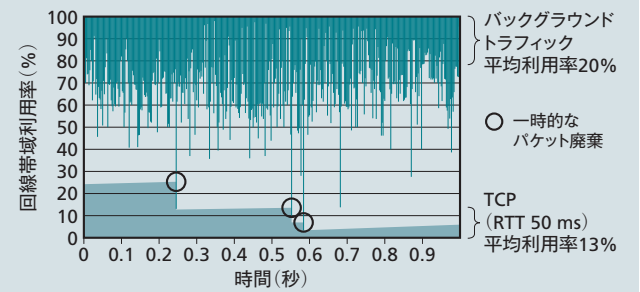
輻輳ウィンドウサイズは，往復通信遅延時間であるRTT（Round Trip Time）に送信可能なデータ量の上限に相当することから，RTTの増加につれて通信帯域が減少してしまう（**図1**参照）。送受信バッファを大きくすることで改善できるが，送受信バッファは有限であり，RTTによっては通信帯域が劣化する。

一方，送信端末は，RTTごとのデータ量を調整するが，パケットごとの送信間隔を制御せず，輻輳ウィンドウサイズに相当するパケットをバースト的（連続的）に送信してしまう（**図1**参照）。この連続的なパケットが，ネットワークの通信機器に他コネクションのパケットと同時に入力すると，通信帯域の低下を招くパケットの廃棄を引き起こす。HTB[5)]，PSPacer[6)]など通信帯域をパケットごとに平滑化する技術が知られているが，通信帯域を一定値に制御する機能であり，動的に通信帯域が変化するTCPの送信データの平滑化に適用することは難しい。

2.2　輻輳制御

輻輳制御は，ネットワークの輻輳状況から通信帯域を判定し，帯域制御による通信帯域の制限を行う機能である。パケット廃棄およびRTTの増減より輻輳状況を判定して制御する手法が存在する。

パケット廃棄による輻輳制御としてCUBIC[1)]，Reno[3)]，HighSpeed[7)]，BIC[8)]，Scalable-TCP[11)]，NewReno[14)]が知られている。これらは，早期の輻輳回復が予想される1 ms以下といった一時的なパケット廃棄時にも輻輳ウィンドウサイズを減少させ，通信帯域を低下させてしまう課題がある。



注：略語説明　TCP（Transmission Control Protocol）

図2｜一時的なパケット廃棄
バックグラウンドのトラフィックとTCPのパケットの送信が競合し，一時的なパケット廃棄が発生する。

一般的なネットワークでは，複数の通信が通信回線を共有する。秒単位の長期的な回線帯域の利用率が50％以下であっても，1 ms以下の短期的な利用率は0％～100％で大きく変化する。**図2**では，長期的な利用率が13％のTCPと20％のバックグラウンドトラフィックが同一の回線を共有しているケースを示している。バックグラウンドトラフィックは短期間の利用率が0％～100％の間で大きく変動しており，TCPとのトラフィックの総和が通信回線の回線帯域を超過し，一時的なパケット廃棄が発生する。

従来の輻輳制御では，1パケットの廃棄が発生するとその発生を検知し，輻輳ウィンドウサイズ，すなわち通信帯域をその直前の値より一定の割合減少させる（**図3**参照）。例えば，Reno[3)]では50％減少させる。ネットワークの回線帯域の長期的な利用率が100％に到達していなくても，バックグラウンドのトラフィックとの競合により，パケットが廃棄され，通信帯域が回線帯域よりも小さい領域で安定し，通信帯域が劣化する。

さらに，通信帯域の低下に加え，複数コネクション間の通信帯域が不公平に分配される課題がある。TCPでは，廃棄がないとRTTごとに一定の帯域を増加させていく（**図3**参照）。RTTが小さなコネクションが輻輳ウィンドウサイズを大きくしやすく，RTTが大きなコネクションの通

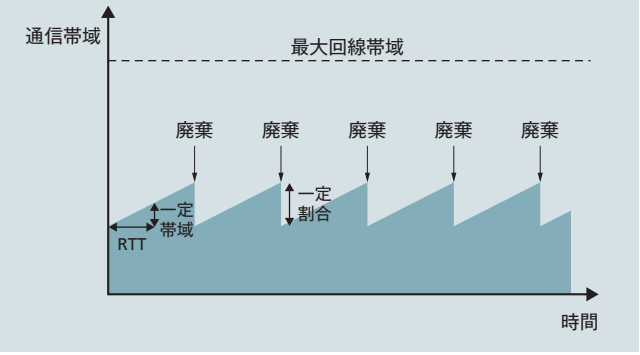


図3｜TCPにおける通信帯域の時間変化
TCPでは，1パケットの廃棄が発生すると一定割合通信帯域を低下させる。一方，廃棄が発生しないと，RTTごとに一定帯域を増加させる。

信帯域が枯渇してしまう。

一方，RTTの増減に基づく輻輳制御として，Vegas[9]，FAST TCP[10]などがある。これらは，輻輳時にネットワーク内の通信機器のバッファに多くのパケットが蓄積されRTTが増加することに着目し，輻輳状況を把握する。パケット廃棄が頻繁に発生するケースではACKが返信されずにRTTを計測できない。さらに，一時的なRTTの増加により輻輳を検出するとウィンドウサイズをあらかじめ定めた割合で単純に減少させる。パケット廃棄による制御と同様に，一時的なパケットの廃棄が発生する通信環境では，通信帯域が劣化してしまう。

パケット廃棄とRTTの両方を利用する輻輳制御も存在するが[2]，同様の理由から一時的な廃棄により，通信帯域が低下しやすい。

2.3　再送制御

TCPの再送制御では，データ欠落のないパケット送信を行うため，廃棄されたパケットの再送信を行う。送信端末はパケットにバイト単位のシーケンス番号を付与して送信し，受信端末はパケットを受信すると次に受信すべきシーケンス番号をACKに記載して送信する。受信端末はACK内のシーケンス番号が変化しないことでパケットの廃棄を認識し，廃棄されたパケットを再送信する。

この制御ではRTTに送信したパケットのうち1パケットの廃棄だけを一往復のデータ送受信により認識可能であり，複数パケットの廃棄が発生するケースでは，パケットの再送信が遅延してしまう。TCPのSACK（Selective ACK）オプション[12]では，受信端末から送信端末へ不連続な受信箇所の最初と最後のシーケンス番号をTCPのオプションフィールドに3か所まで記載して通知する（**図4**参照）。送信端末は，受信箇所から廃棄されたデータを判定し，最大4か所の再送信を行う。同図は，送信したデータA～Lのうち，B，D，F，H，Jが廃棄されるケースを記載している。受信したデータがC，E，Gであることを通知することで，送信端末はB，D，F，Hの廃棄を認識するが，データJが廃棄されたことを認識できない（**図4**参照）。RTT後に受信するACKにより，このデータを認識するため再送信が遅延する。実装によっては通知可能なデータ箇所を増加させることも可能だが，最大数に制限があり，通信帯域の低下を招く。

3.　高速TCPの開発

TCPの帯域制御，輻輳制御，再送制御の課題を解決する高速TCPを提案した。以下では，その詳細について述べる。

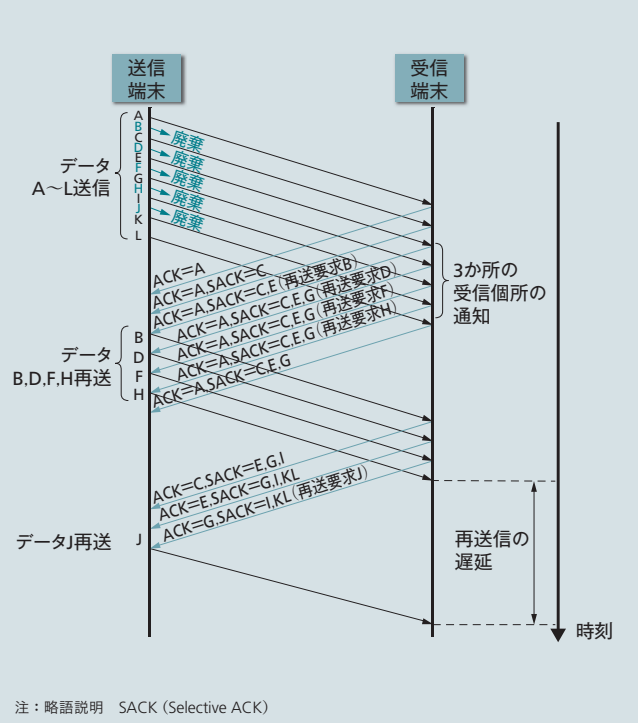


図4｜SACKを用いた再送制御

TCPのSACKオプションでは，廃棄箇所を4か所まで再送信が可能である（データB, D, F, H）。廃棄箇所が5か所以上となると，再送信が遅延する（データJ）。

3.1　トークンバケットを用いた帯域制御

輻輳ウィンドウサイズに代わってトークンバケットアルゴリズムによる帯域制御を行う。送信端末は送信可能なデータ量（byte）に相当するトークンを蓄積するトークンバケットを管理する。単位時間ごとに通信帯域に比例したトークンが蓄積され，次に送信するパケットのバイト長に相当するトークンが蓄積されていると，パケットを送信する。後述の輻輳制御で判定される通信帯域に応じて蓄積するトークン量を変化させることで通信帯域を制御可能である。

この制御では，RTTと独立した帯域制御が可能であり，RTT増加による通信帯域の低下を抑止可能である。さらに，トークンバケットの大きさにより，連続送信するパケットを制限することができ，バースト転送によるパケット廃棄を防止できる。

3.2　廃棄率に基づく輻輳制御

提案する輻輳制御は，一時的に発生する廃棄と回線帯域の定常的な枯渇による廃棄を識別し，通信帯域を判定することを特徴とする。

1 ms以下の一時的な廃棄であれば，10 ms以上といった広域網で一般的なRTTにおける廃棄率の増加は小さいことに着目した。直前のRTTの間の廃棄率の変化率が一定値以下の場合には一時的な廃棄と判定して通信帯域を増加させ，一定値を超過する場合には，定常的な枯渇による廃

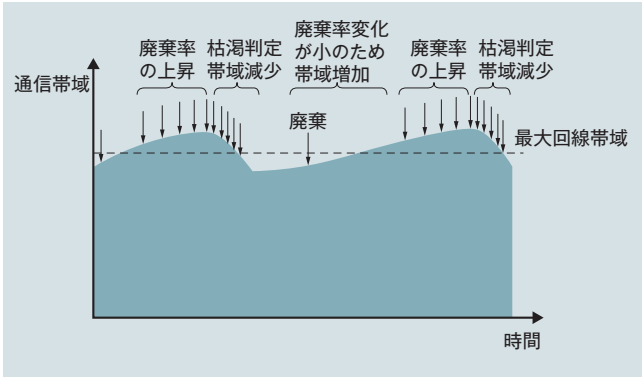


図5｜廃棄率の変化率に基づく輻輳制御

高速TCPの輻輳制御では，廃棄率の変化により，一時的に発生するパケット廃棄と定常的な帯域枯渇による廃棄を識別し，通信帯域を判定する。

棄と判定し，通信帯域を低下させる（**図5**参照）。

廃棄率の計算では，まず，送信端末は後述のNACK（Negative ACK）により，直前のRTTの間に廃棄されたデータ量を求め，廃棄データ量をRTTで除算することで廃棄帯域（超過帯域）を計算する。次に，廃棄帯域をRTT前の通信帯域で除算することで廃棄率を計算する。

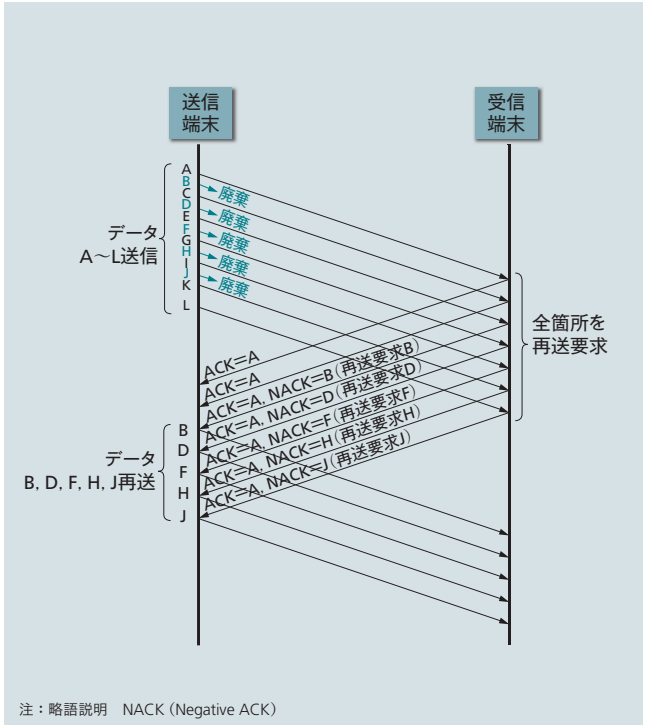
廃棄率の変化率が一定値を超過した際，RTT前の通信帯域より廃棄帯域を減算した値を新たな通信帯域とする。一方，廃棄率の変化が一定値以下の場合には，一時的な廃棄と判定して通信帯域を増加させる。この増加量をRTT前の通信帯域が小さいほど，増加量を大きくすることで，コネクション間の通信帯域の不公平な分配を抑止する。

この提案の輻輳制御は，定常的に回線帯域が枯渇するときに限定して通信帯域を減少させるため，通信帯域の低下を抑止し，回線帯域を使い切ることができる。また，通信帯域の増加時には通信帯域が小さなコネクションに優先的に通信帯域を分配することで，公平な分配を実現する。

3.3　NACKを用いた再送制御

提案する再送制御では，到着箇所を通知するSACKの代わりに不連続な廃棄箇所を送信端末に通知するNACK[13]を用いる。ここでNACKを用いたのは，廃棄箇所を直接通知することで，送信端末の計算負荷を低減するためである。

この制御では，SACKと同一のTCPオプションフィールドを使用し，各廃棄箇所を1回だけ通知する。廃棄箇所が4か所よりも多い場合にも，受信端末はすべての廃棄箇所を1回のパケットとACKのやり取りで送信端末へ通知可能であり，送信端末は，全廃棄箇所を迅速に再送信できる（**図6**参照）。廃棄検出から再送信完了までの時間が短縮されるため，再送信の遅延による通信帯域の低下を抑止できる。



注：略語説明　NACK（Negative ACK）

図6｜SACKを用いた再送制御

高速TCPの再送制御では，廃棄箇所を1回のみ通知することで，全廃棄箇所の迅速な再送信を実現する。

表1｜TCPと高速TCPの比較

高速TCPでは，帯域制御，輻輳制御，再送制御における課題を解決する手法を採用した。

		帯域制御	輻輳制御	再送制御
TCP	手法	• ウィンドウサイズによる制御	• 1パケットの廃棄で，送信帯域を定率減少	• SACKにより廃棄箇所を最大4か所通知
	課題	• RTTの増加に伴い帯域低下 • バーストトラフィックの送信	• 一時的な廃棄発生による帯域低下	• 再送が遅れ帯域低下
高速TCP	手法	• トークンバケットによる制御	• 廃棄率の変化に基づき送信帯域を制御	• NACKにより廃棄箇所をすべて通知
	効果	(1) RTTに非依存な帯域制御 (2) 平滑化されたトラフィックの送信	(3) 一時的な廃棄発生時の通信帯域低下の抑止	(4) 廃棄パケットを最短でRTT時間以内に再送

3.4　方式比較

TCPと提案の高速TCPの比較を**表1**に示す。帯域制御，輻輳制御，再送制御のそれぞれに対応した課題を解決する手法を提案している。以下では，提案の高速TCPを，実環境を模擬した評価環境を用いてTCPとの比較評価を行い，その効果を実証する。

4.　高速TCPの評価

4.1　RTT依存性の評価

帯域制御による効果を確認するため，通信帯域のRTT依存性を評価した。

評価構成を**図7**に示す。端末#1，#2とTCP(CUBIC-TCP[1])

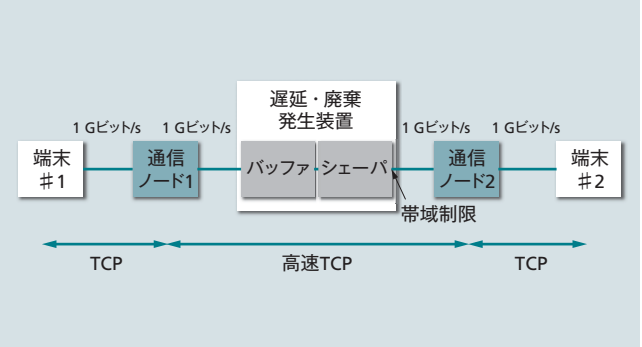


図7｜評価構成

広域網を模擬する遅延・廃棄発生装置を用いる評価構成とした。TCPと高速TCPの変換機能を通信ノード1，2に搭載し，広域網通信に相当する通信ノード間の通信を高速化した。

と高速TCPを変換する機能を備える。通信ノードおよび広域網を模擬する遅延・廃棄発生装置を1 Gビット/sの回線で接続した。端末＃1と通信ノード1，端末＃2と通信ノード2間は従来のTCPによる通信，通信ノード1と2間は高速TCPによる通信となる。通信ノードの変換機能をオフとすると端末＃1，＃2間はすべて従来のTCPによる通信となる。遅延・廃棄発生装置はパケットの遅延時間と廃棄率を制御する機能に加え，通信ノード2への通信帯域を制限するシェーパとバッファを備えている。

RTTは日本―米国間で100 ms～200 ms，日本―ブラジル間で300 ms程度であり，多くの通信が500 ms以下と想定されるため，遅延・廃棄発生装置で最大500 msの遅延を発生させた。また，廃棄率を0.01%，送受信バッファ量を16 MB，シェーパの帯域制限をオフとして送受信端末間で1 GBのファイルを端末＃1から＃2にFTP（File Transfer Protocol）により転送した。評価結果を**図8**に示す。

CUBIC-TCPは，RTTが国内相当の10 msの際には255 Mビット/sとなるが，日本―米国間相当の200 msの際には16.5 Mビット/sに大きく低下する。一方，高速TCPでは，RTTが大きくなっても，送受信バッファより定まる理論限界値（送受信バッファ量／RTTと回線帯域の小さな値）に近い通信帯域を達成している。例えば，200 msの際には理論値707 Mビット/sに対して390 Mビット/sとなり，TCPの23.6倍の通信帯域を達成している。この評価により，高速TCPがTCPと比較してRTTに依存しない通信帯域を実現できることが分かる。

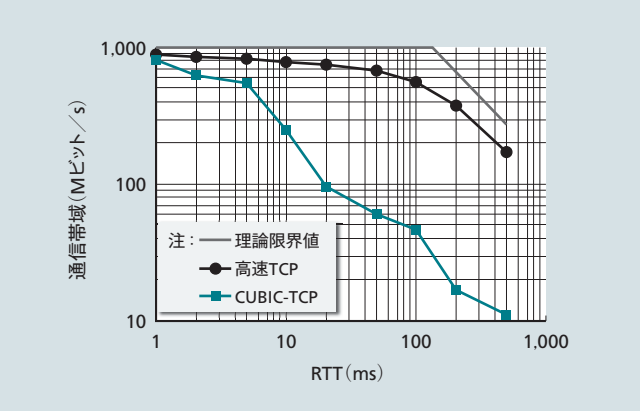


図8｜RTTと通信帯域の関係

CUBIC-TCPでは，RTTの増加に伴い通信帯域が大きく低下するのに対し，高速TCPでは，通信帯域の低下は小さく，理論限界値に近い通信帯域を達成した。

4.2　トラフィックの平滑化効果の評価

トークンバケットアルゴリズムによるトラフィック平滑化の効果を評価した。廃棄が発生しやすい評価環境として平滑化の効果が分かるように，**図7**のシェーパによって100 Mビット/sに通信帯域を制限した。トークンバケットの大きさを12 KB，RTTを200 ms，廃棄率を0，送受信バッファ量を16 MBとし，シェーパ前段のバッファ量を変化させた。評価結果を**図9**に示す。

CUBIC-TCPは，輻輳ウィンドウサイズに基づく送信制御により，RTTごとにバースト的なトラフィックを発生させるため，遅延・廃棄発生装置のバッファ量が小さいと廃棄が発生し，通信帯域が劣化している。例えば，64 KBの際には7.8 Mビット/sまで低下している。

一方，高速TCPでは，バッファ量より小さな12 KBのトークンバケットを用いているため，パケットのバースト送信は最大で12 KBとなり，バッファ量が64 KB以上の評価環境においては，バッファ量によらず80 Mビット/s～90 Mビット/sの安定した通信帯域が得られた。上述のように，提案の帯域制御はバースト的なパケット送信を抑止することで，通信装置のバッファ量が小さなときにもパケットの廃棄を防止し，通信帯域を向上できることが分かる。

4.3　パケット廃棄の評価

輻輳制御の効果を評価するため，通信帯域の廃棄率依存

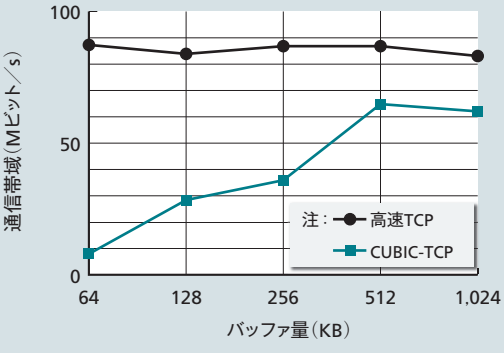


図9｜バッファ量と通信帯域の関係

CUBIC-TCPでは，バッファ量が小さいと通信帯域が低下するのに対し，高速TCPでは，トラフィックの平滑化の効果によりバッファ量に依存しない通信帯域を実現した。

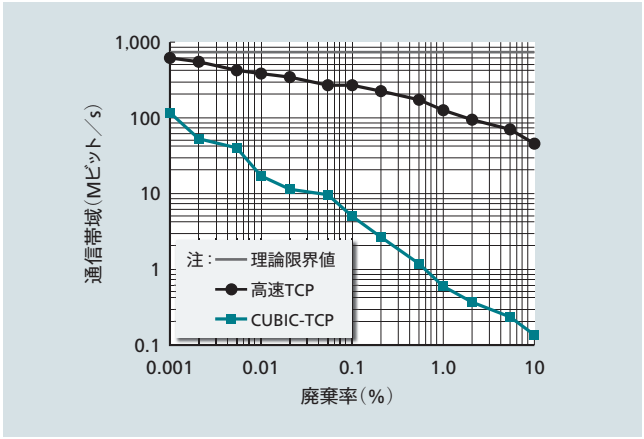


図10｜廃棄率と通信帯域の関係

CUBIC-TCPでは，廃棄率の増加に伴い通信帯域が大きく低下するのに対し，高速TCPでは，通信帯域の低下は小さく，廃棄率1.0%において134 Mビット/sの通信帯域を達成した。

性を評価した。**図7**の評価構成にてRTTを200 msとし，廃棄率を変化させた。廃棄率と通信帯域の関係を**図10**に示す。

CUBIC-TCPの通信帯域は，廃棄率0.001%では111 Mビット/sとなるが，廃棄率の増加とともに急激に低下し，インターネットの廃棄率として想定される1.0%では0.6 Mビット/sに低下する。一方，高速TCPでは，1.0%においても134 Mビット/sとなり，CUBIC-TCPに対し，2桁以上高い通信帯域を実現している。

次に廃棄率を1.0%として，送受信バッファ量依存性を評価した。結果を**図11**に示す。CUBIC-TCPの通信帯域は，送受信バッファ量，すなわち輻輳ウィンドウサイズの最大値を大きくしても向上せず，0.6 Mビット/s程度の性能となる。**図7**の評価構成では，通信帯域のボトルネックがなく，バースト送信による廃棄がない。このため，CUBIC-TCPの通信帯域低下は，廃棄発生時の輻輳制御，再送制御に起因する。

一方，高速TCPは，送受信バッファ量を大きくすることで通信帯域が134 Mビット/sより708 Mビット/sまで向上した。輻輳ウィンドウの制限を無くすことで，高速TCPの輻輳制御，再送制御が廃棄発生時の課題を解決し，通信帯域を向上する。

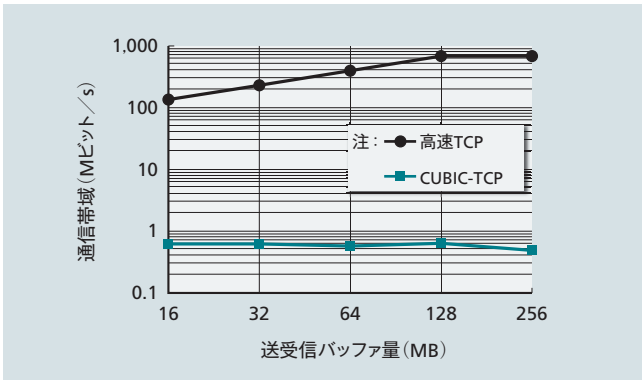


図11｜送受信バッファ量と通信帯域の関係

CUBIC-TCPでは，通信帯域が常に低くなるのに対し，高速TCPでは，送受信バッファ量が大きくなるにつれて輻輳ウィンドウサイズの最大値が大きくなり，通信帯域が向上した。

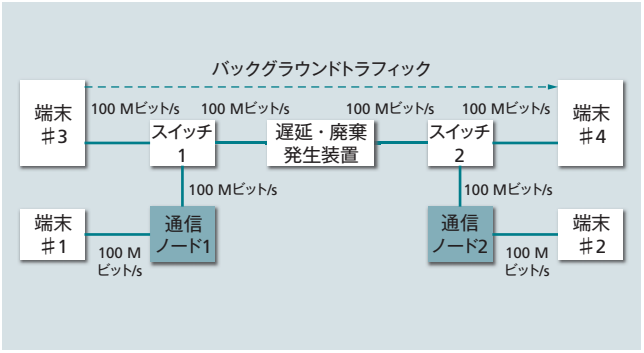


図12｜評価構成

高速TCPのバックグラウンドにトラフィックを挿入する構成とした。端末#3，#4間でバックグラウンドトラフィックを発生させ，端末#1，#2間の通信帯域を測定した。

4.4　余剰帯域の効率活用

輻輳制御による余剰帯域の効率活用の評価を行うため，高速TCPのバックグラウンドにトラフィックを挿入し，その帯域を変化させた。評価構成を**図12**に示す。

TCPと高速TCPを変換する通信ノード1，2，遅延・廃棄発生装置，端末#1～#4およびスイッチ1，2が100 Mビット/sの回線で接続される。

端末#3から#4に向けて50 Mビット/sのパケット送信と停止を1秒ずつ繰り返すバックグラウンドトラフィックを挿入し，端末#1から#2への通信帯域の時間変化をTCP変換の有無を切り替えて測定した。遅延・廃棄発生装置によるRTTを200 ms，廃棄率を0とした際の結果を**図13**に示す。

CUBIC-TCPは，バックグラウンドのトラフィックとの競合による廃棄により，通信帯域が低下しやすく，平均

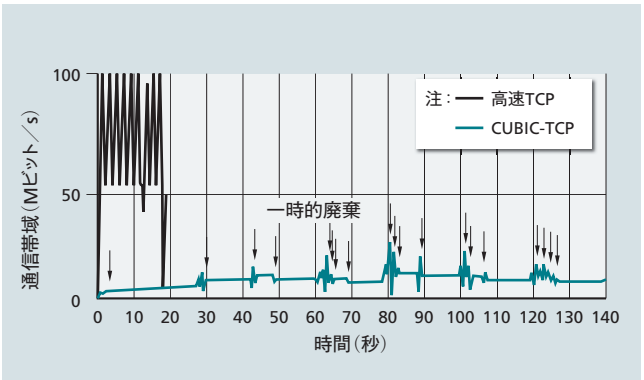


図13｜余剰帯域変化時の通信帯域の時間変化

CUBIC-TCPでは，一時的な廃棄により通信帯域が低下するのに対し，高速TCPでは，バックグラウンドトラフィックの余剰帯域の効率活用（余剰帯域の91%を利用）が可能であった。

75 Mビット/sの余剰帯域のうち10 Mビット/s（13%）の使用にとどまった。一方，提案する高速TCPでは，68 Mビット/s（91%）に向上した。高速TCPの輻輳制御が余剰帯域を検出し，その効率活用を図ることができる。

4.5　コネクション間の通信帯域の分配

複数コネクションが回線帯域を共有する際の通信帯域の分配を評価した。

この評価では，高速TCPを実装した端末＃1～＃4，遅延・廃棄発生装置，スイッチ1, 2を100 Mビット/sの回線で接続する構成とした（**図14** 参照）。端末＃1，＃2間と端末＃3，＃4間に高速TCPのコネクション1と2を設定し，コネクション1のRTTを200 msとし，コネクション2のRTTを変化させた。146 MBのファイルをFTP転送した際の各コネクションの通信帯域を**図15**に示す。

コネクション2のRTTが変化しても，各コネクションの帯域は回線帯域の50%に近い45.1 Mビット/s以上の通信帯域を確保している。通信帯域の小さなコネクションに優先的に通信帯域を分配するため，RTTに大きく依存しない通信帯域を実現したと考えられる。

提案の輻輳制御は，通信帯域の向上に加え，コネクション間の通信帯域の公平な分配を実現できることが分かる。

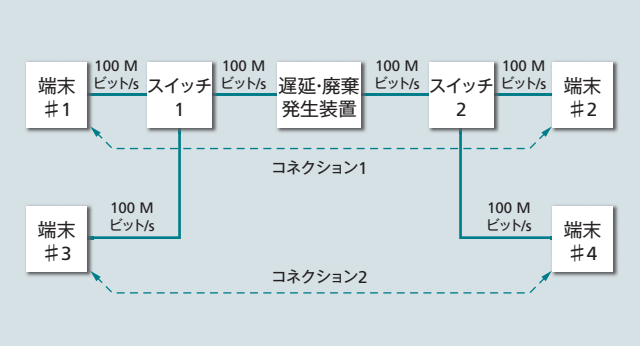


図14｜評価構成

RTTの異なるコネクションを設定する構成とした。端末＃1，＃2間のコネクション1のRTTを200 msとし，端末＃3，＃4間のコネクション2のRTTを変化させ，それぞれの通信帯域を測定した。

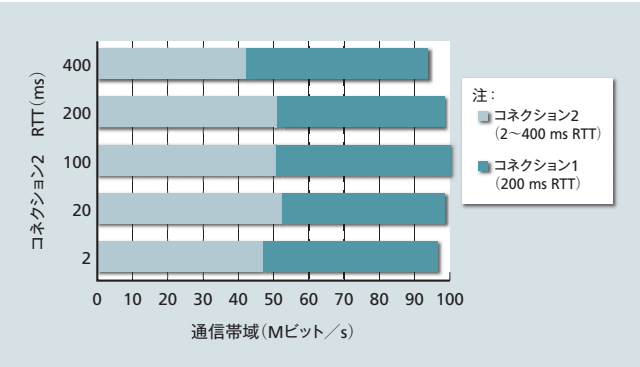


図15｜コネクション間の通信帯域分配

各コネクションに通信帯域が公平に分配され，100 Mビット/sの回線帯域のうち，45.1 Mビット/s以上の通信帯域が各コネクションに分配された。

5.　高速TCPのユースケース

高速TCPのユースケースとしてデータバックアップやプライベートクラウドなどを想定している。以下，その詳細について述べる。

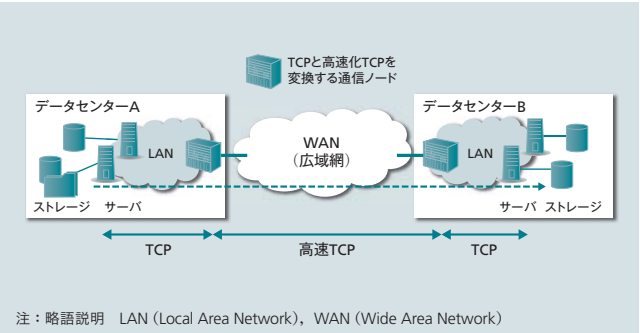
5.1　データバックアップ

データバックアップでは，広域災害に対応するため，例えば，東京，九州といった遠隔の拠点間でデータを転送する必要がある。データ量はギガバイト以上の大容量データであることが多く，例えば，企業の全データを夜間にバックアップする際には，通信帯域の低下によって業務開始時間までにバックアップが完了しない状況が発生する。

このユースケースでは，データセンターなどの拠点にTCPと高速TCPの変換機能を搭載した通信ノードを配置する（**図16**参照）。通信遅延が大きな広域網の高速化が可能なため，結果としてストレージ間の通信の広帯域化を実現可能である。

5.2　プライベートクラウド

ユーザーの各拠点およびデータセンターに前述の通信ノードを配置することで広域網の高速化を実現する（**図17**参照）。通信ノードがTCPと高速TCPの変換機能を備えることで，ユーザーのコスト負担を強いる端末のTCP変更が不要となる。アプリケーションによっては少量のデー



注：略語説明　LAN（Local Area Network），WAN（Wide Area Network）

図16｜データバックアップへの適用例

高速TCPを用いて，データセンターAの大容量データをデータセンターBにバックアップする。

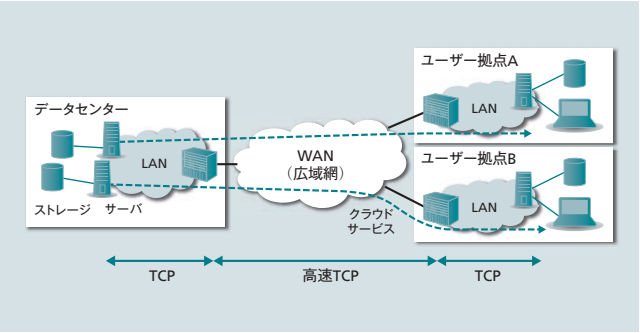


図17｜プライベートクラウドへの適用例

高速TCPにより，ユーザー拠点AおよびBのユーザーに対し，高応答な各種クラウドサービスを提供する。

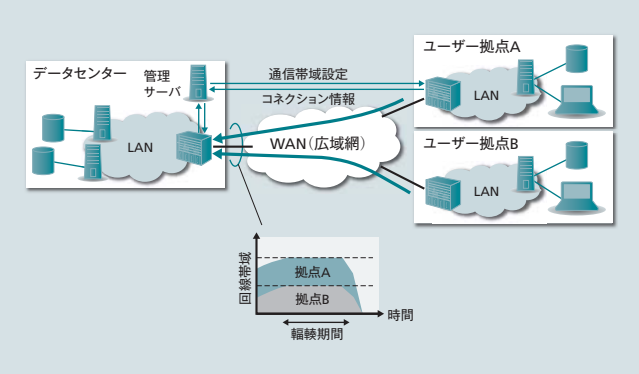


図18｜帯域制御機能と統合運用管理
ユーザー拠点A，Bからデータセンターへのアクセスを自動的に調停し，通信帯域を制御する。

タを転送するケースもあり，効果が小さいこともあるが，設計データのような大容量データを送受信するケースでは，前述した効果を見込める。

6. 今後の研究開発

今後は，高速TCPによる通信帯域の実現と，ネットワーク管理の容易化を両立する技術の開発を推進する。

例えば，複数拠点からデータセンターにアクセスが集中する**図18**のケースでは，通信帯域を確保するために，各拠点からのアクセスを調停する必要がある。高速TCPの通信帯域の制御機能とネットワーク管理者のポリシーなどに基づく本機能の自動設定，制御機能が必要となる。

高速TCPのクラウド適用と並行して，高速TCPのクラウド適用時の運用管理を効率化する技術開発を推進していく。

7. おわりに

ここでは，クラウド適用に向けた高速TCPとその評価実験について述べた。

実環境を模擬した多様な通信環境下において，提案した帯域制御，輻輳制御，再送制御の有効性を示した。例えば，日本―米国間の通信を想定したRTTが200 ms，パケット廃棄率が0.1％の通信環境で，23.6倍の通信帯域を達成した。

今後，クラウドコンピューティングへの高速TCPの適用を推進するとともに，高速TCPにおける通信帯域の制御機能，その統合運用管理技術の開発を進めていく。

参考文献など

1) I. Rhee, et al. : CUBIC: A new TCP-Friendly high-speed TCP variant, Third International Workshop on Protocols for Fast Long-Distance Networks , (2005.2)
2) K. Tan, et al. : A Compound TCP Approach for High-speed and Long Distance Networks, IEEE, INFOCOM 2006 (2006.4)
3) M. Allman, et al. : TCP Congestion Control, IETF, RFC2581 (1999.4)
4) J. Postel : Transmission Control Protocol, IETF, RFC793 (1981.9)
5) HTB，http://luxik.cdi.cz/~devik/qos/htb/
6) GridMPI，http://www.gridmpi.org/
7) S. Floyd : HighSpeed TCP for Large Congestion Windows, IETF, RFC3649, (2003.12)
8) L. Xu, et al. : Binary increase congestion control (BIC) for fast long-distance networks, IEEE, INFOCOM 2004 (2004.3)
9) L. Brakmo et al. : TCP Vegas: End to End Congestion Avoidance on a Global Internet, IEEE journal on selected areas in communications, Vol. 13, No.8 (1995.10)
10) C. Jin, et al. : FAST TCP: motivation, architecture, algorithms, performance, IEEE, INFOCOM 2004 (2004.3)
11) T. Kelly : Scalable TCP: Improving performance in highspeed wide area networks, ACM SIGCOMM Computer Communication Review (2003.4)
12) M. Mathis, et al. : TCP Selective Acknowledgment Options, IETF RFC2018 (1996.10)
13) J.P. Macker, et al. : A TCP friendly, rate-based mechanism for NACK-oriented reliable multicast congestion control, IEEE, GLOBECOM 2001, Vol.3, pp.1620-1625 (2001)
14) S. Floyd, et al. : The NewReno Modification to TCP's Fast Recovery Algorithm, IETF, RFC2582 (1999.4)

執筆者紹介

矢崎 武己
1995年日立製作所入社，中央研究所 情報システム研究センタ ネットワークシステム研究部 所属
現在，クラウドネットワークの研究開発に従事
電子情報通信学会会員，情報処理学会会員

磯部 隆史
2003年日立製作所入社，中央研究所 情報システム研究センタ ネットワークシステム研究部 所属
現在，広域ネットワーク高速化に関する研究開発に従事
電子情報通信学会会員