解説

# 長距離高速通信のための TCP 性能改善技術の動向

鶴　正人　九州工業大学
tsuru@cse.kyutech.ac.jp
熊副和美　通信・放送機構 / 日本テレコムインフォメーションサービス
kuma@genesis.tao.go.jp
尾家祐二　九州工業大学
oie@cse.kyutech.ac.jp

広域 IP 網の大容量化に伴って Grid に代表されるような長距離高速通信を要求するアプリケーションが現実のものとして登場し，その効率的実現のための研究開発が活発化している．本稿は，長距離広帯域パス上での通信，特に現在の IP のアプリケーションが標準的に使うトランスポートプロトコルである TCP による大容量データ転送の問題点を説明し，長距離高速通信を可能とするトランスポートプロトコルの実現のための研究動向および日本における JGN 上に構築された長距離・大容量テストベッド上での予備的実験を紹介する．

## はじめに

広域 IP 網の大容量化（たとえば，10Gbps の帯域幅を持ったバックボーンネットワーク）に加え，LAN および端末の大容量化（10GbE のインタフェースを持ったスイッチや NIC の登場）の実現によって，エンドツーエンドでギガビットレベルの帯域幅を持ったパス（ネットワーク上のエンドツーエンドの経路）が利用可能になってきた．そして，データ Grid に代表されるような広域 IP 網上で大容量データ転送を必要とするアプリケーションの出現により，たとえば，大陸を跨いで数テラバイトのファイルを高速に転送したい，というような要求が発生してきた．

ここで，「高速」とは，「スループット」（単位時間あたりのエンドツーエンドでの転送データ量）が大きいことを意味する．すなわち，大きなデータを多数の IP パケットに分割して転送する場合に，パケットロス発生時の再送も含めて，いかに短時間でデータ全体を転送できるかという，トランスポートプロトコルから見た性能の問題である．そして，そのような「高速」通信は，ネットワーク回線やルータの性能（帯域幅）向上だけで実現できるものではなく，「長距離」となるとさらにやっかいな問題が発生する，という話が本稿のテーマである．たとえば，米国 Internet2 [☆1] プロジェクトが主催する Land Speed Record（LSR）[☆2] という TCP 長距離高速通信の性能を競うコンテストがあるが，その評価尺度は距離×スループットであり，距離が長いと困難度が増すことを暗示している．ちなみに現時点での LSR の IPv4 部門での記録は 238888 テラビットメータ／秒で，これは 1.1 テラバイト（TB）のデータを 2.5Gbps の帯域幅を持つ 10037km（米国－欧州間）のパス上で 3700 秒かかって転送したものであり，平均スループットは 2.38Gbps であった．

IP 通信においてエンドツーエンドのデータ転送を提供するトランスポートプロトコルとしては，TCP と UDP があるが，WWW, ファイル転送，メール配送等，1 ビットの不足もなく完全なデータを転送する必要があるアプリケーションの多くは TCP を利用している．これは，TCP が，パケットの転送結果を保証しない IP 通信網の上で信頼性のあるエンドツーエンドのデータ転送を提供するために，次のような機能を有しているからである；(1) 誤り制御：パスの途中でパケットの廃棄（パケットロス），データ誤り，順序逆転が発生した場合に，再送や並べ替えを行う；(2) エンドツーエンドのフロー制御：受信側の処理能力以上にパケットが到着しないように，適応的に送信速度を調整する；(3) 輻輳制御：ネットワーク状態を推定して，適応的に送信速度を調整する（パスの途中が空いているようであれば速度を上げ，混んでいるようであれば速度を落とす）．

さて，高速通信の話に戻る．パスの途中のリンクやル

☆1 http://www.internet2.edu/
☆2 http://lsr.internet2.edu/

ータの性能が十分あり，そのパスに沿って，ある巨大な帯域幅が利用可能であるような状況において，その帯域幅ぎりぎりの高スループットを実現するためには，さまざまな要素を考慮する必要がある．

### 送信側，受信側のエンドホストの処理能力

NIC の送受信能力，内部バスのデータ転送速度，プロトコル処理を行うための CPU 能力，ディスクやメモリのアクセス速度・容量等が目標スループットに見合うことが不可欠である．また，OS 内部のバッファ長等も標準設定では不十分な場合もある．高速化のために，パケット（データ）のコピーを最小限に減らす方式や，プロトコル処理をハードウェア化する方式等も研究開発されている．内部バスは PCI-X 等による高速化が進んでいる．

### プロトコルヘッダのオーバーヘッド

たとえば TCP を用いる場合，1 個の IP パケットに付き，IP ヘッダ 20 バイト，TCP ヘッダ 20 ～ 40 バイトが無駄に消費される．しかし，これは，IP パケット内の TCP データ長が十分に大きければ相対的に無視できる．つまり，ヘッダ部分の割合が小さくなればよい．あるリンク上で転送可能な最大の IP パケット長を Maximum Transfer Unit（MTU）と呼び，イーサネットの場合は 1.5 キロバイト（KB）が標準的に用いられるが，最近の高速リンクでは 9KB 等の大きな値も利用可能になっている．もちろん，パス上のすべてのリンクの MTU が大きいことが必要である．

### 帯域幅を埋める「適切な」送信速度の調整

まず，(i) 自分専用にその帯域幅を割り当てられており，専有できる；(ii) 他の通信とその帯域幅を共有して使う；の 2 つの場合に分けて考える必要がある．(i) の場合は，専有できる帯域幅を事前に知っていれば，その帯域幅ぎりぎりの一定速度（一定間隔）でパケットを送信すればよい．しかし，通常の UNIX や Windows のような OS では，指定速度でパケット送信を維持することは，高速であるほど困難になり，最小帯域幅を持つリンクの手前のルータ等でのバッファ溢れ（パケットロス）の発生は避けられない．また，その帯域幅を知らない場合には，それを推定する手法も必要になる．(i) の場合を想定した UDP ベースの高速データ転送プロトコルとして，Reliable Blast UDP（RBUDP）[☆3] や TSUNAMI[☆4] が提案されている．

一方，(ii) の場合は，自分が使える帯域幅は他人の挙動に依存して時間とともに変化し，また，他人を押し退けて自分の使う帯域幅を増やすという手段もあり得る．よって，大きな変化への追従性や帯域幅配分の公平性といった尺度で検討する必要がある．たとえば，パスの空き（可用帯域幅）を素早く感知し送信速度を上げること，その時に他の通信に悪影響を与えないこと，可用帯域幅を奪い合う利用者間の公平性が高いこと，ネットワークが輻輳した時に素早く送信速度を落とせること，その際の身の引き具合いに関しても利用者間の公平性が高いこと等が目標になり，これらは TCP の目標とも合致する．以降では，送信速度の調整に関して，(ii) の状況で考えよう．

## 長距離高速通信における TCP の問題

信頼性のあるトランスポートプロトコルは，何らかのかたちの誤り制御や輻輳制御を行う．そして，高速な通信になるほど短い時間スケールでの迅速な誤り制御や輻輳制御が必要になるが，一方では，長距離になるほど伝播遅延によりフィードバックが遅れる．よって，長距離通信において，エンドツーエンドのプロトコルを用いた制御を適切に行って高速性を実現することは，本質的な困難を抱えている．

### TCP のウィンドウ制御

ここでは，TCP における送信速度の調整が，誤り制御・フロー制御・輻輳制御が渾然一体となったウィンドウ制御として実現される様子を説明する．まず，略語を定義する．

**RTT（Round Trip Time）**：エンドホスト間で IP パケットが往復するのに必要な時間．エンドツーエンドプロトコルの進行の基本単位時間になる．TCP はタイムアウト時間の決定やその他の制御のために RTT を継続的に計測して把握する必要がある．なお，最小値は距離（伝搬遅延）に依存するが，そこからの増分・変動はパスの途中の混雑を反映する．

**MSS（Maximum Segment Size）**：IP パケットが運ぶことができる最大の TCP データ長．MTU から IP ヘッダ長と TCP ヘッダ長を引いた値．

**BDP（Bandwidth-Delay Product）**：パスの帯域幅と RTT の積（帯域幅遅延積）．あるデータを送った後，それが正しく届いたことの確認が受信側から戻ってくるまでに送信可能な最大のデータ量．

以降の説明では，TCP プロトコル定義（RFC793）に準拠するだけでなく，RFC2581（TCP Reno）に基づいたスロースタート，輻輳回避，高速再送，高速回復

☆3 http://www.evl.uic.edu/cavern/quanta/
☆4 http://www.indiana.edu/~anml/anmlresearch.html

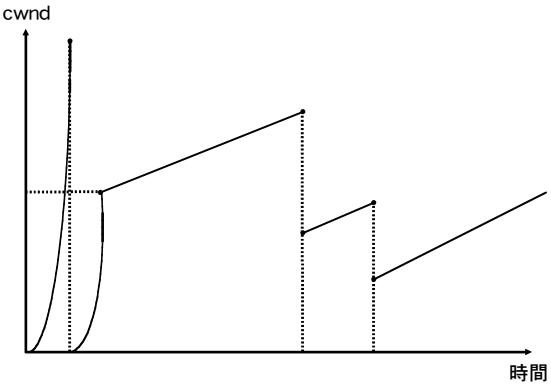


図-1　輻輳ウィンドウ長の時間変化

と，RFC2018 および RFC3517 に基づいた選択的受信確認（SACK）が実装されている場合を想定する．

1. 送受信される TCP データはバイト単位のシーケンス番号（SN）で識別され，受信側は，それ以前に正しい順序で到着した最後のデータの SN の次の番号を，受信確認番号（AN）として送信側へ返す．この確認通知の情報を ACK と呼ぶ．送信側は，対応する ACK を受け取るまでは，再送の可能性に備えて送信データを保持する．以降では簡単のために，ACK は，送信 IP パケットと 1 対 1 にすぐに返される場合を想定する．ただし，現実には多くの場合，2 個以上の IP パケットごとに ACK が返される（「遅延 ACK」）．
2. ここで，送信側は送ったデータに対する ACK を待たずに，次の MSS バイトのデータを次の IP パケットに乗せて送ってよい．送信済みデータの中で ACK を受け取っていない最小の SN から，未送信データのうち現時点で送信が許可される最大の SN までの SN の範囲を，ウィンドウと呼ぶ．各時点でのウィンドウ長は，送信側が輻輳制御の目的で管理する輻輳ウィンドウ長（以降 cw と書く）と，受信側がフロー制御の目的で送信側に伝える広告ウィンドウ長との小さい方の値になるが，以降では簡単のために，広告ウィンドウ長は十分大きいとし，ウィンドウ長 と cw を同一視する．結局，送信速度の調整 ＝ cw の調整であり，時間平均スループット ＝ 時間平均 cw/RTT である．
3. ひとまとまりのパケット列を送信して ACK を待つ場合，もしその中のあるパケット以降のすべてが連続してロスすると，ロスの直前のデータに対する ACK が戻った後は ACK が来ない．この場合，送信側は，ACK の到着をある時間待った後，タイムアウトによってロスを認識し，以降のパケット群を再送する．一方，もしあるパケットだけがロスし，その後のパケットは受信側に届いたとすると，受信側は，最後に受信したデータに対する ACK ではなく，ロスが発生した手前のデータに対する ACK を再送する．この場合，ロスの後の 3 個のパケットが受信側に届くと，送信側へ計 4 回同じデータに対する ACK を返すことになり，送信側は，それら（「3 重複 ACK」）を受け取った時点でロスを認識し，すぐにロスパケットを再送する（高速再送）．
4. cw は，以下の Additive Increase Multiplicative Decrease（AIMD）アルゴリズムに従って調整される．cw の時間変化は典型的には**図-1** のようになる．

**スロースタートフェーズ**：ある初期 cw から出発する．（ある送信パケットに対する）ACK が戻ってくるごとに，cw を MSS バイト増やすので，cw が時間に対して指数的に増加して，急速に送信速度が上がる．その結果，輻輳が起き，一連のパケットが連続的にロスし，タイムアウトが発生する．その場合，スロースタート閾値（ss）を「ロス発生時の cw の半分」に設定し，cw を初期値に戻した後，再び指数的に増やす．cw が ss 以上になると，輻輳回避フェーズに切り替わる．

**輻輳回避フェーズ**：ACK が戻ってくるごとに，cw を（MSS/cw）バイト増やすので，RTT 当たり MSS バイト（IP パケット 1 個分）の増加である．送信速度が緩やかに上がる途中で間欠的なロスに出会い，3 重複 ACK が発生する可能性が高い．その場合，高速再送を行い，再送に対する ACK が戻った後，cw と ss をともに「ロス発生時の cw の半分」に設定する（高速回復）．一方，もし連続的なロスによるタイムアウトが発生した場合は，スロースタートフェーズに切り替わる．

図-1 のような典型的な輻輳回避フェーズ中の平均スループットを見積もってみよう．単純化して，一定周期で 1 個のパケットロスが発生し，cw が 最大値 W と W/2 の間をノコギリ形に変動すると仮定する．

- RTT 秒で cw は MSS バイト増加し，一方 1 周期での cw の増加は W/2 なので，1 周期は RTT*W/(2MSS) 秒．
- 平均スループット S は，3W/4RTT なので，1 周期に転送されるバイト数は，(3W/4RTT)*RTT*(W/2MSS) = $(3W^2)/8MSS$．言い換えると，$(3W^2)/(8MSS^2)$ 個に 1 個しかロスしないことになる．ロス率を p と置けば，$p = (8MSS^2)/(3W^2)$．
- よって，S は p の関数で表現できる：
  $S = (3\sqrt{8}\,MSS)/(4\sqrt{3p}\,RTT) = 1.2MSS/(\sqrt{p}\,RTT)$．

## TCP のスループットが上がらない原因

以上のことから，TCP を長距離広帯域パス上で利用する場合の問題が予想できる．

[1] スループットは cw/RTT で押えられるので，パスの帯域幅を最大限まで使いたいなら，cw を BDP まで

大きくすることが必要になる.

[2] スロースタートフェーズでは，BDP が大きい場合，指数的増加とはいえ，cw が非常に小さい初期値から出発して大きな値に達するまでには時間がかかる．また，そのように非常に大きな cw においてロスが起きる状況では，多量のパケットが連続送信されてネットワークに過度の輻輳を引き起こす.

[3] 輻輳回避フェーズでは，cw の増加速度は MSS/RTT であり，RTT が大きいと非常に遅い．一方，間欠的なロスは，ネットワークがあまり混雑していなくても発生し，その時 cw が半分まで下がるが，BDP が大きい場合にその半分を回復するには長い時間を必要とする．逆に言えば，ロス率が非常に低くないと，平均 cw を大きく保つことができない.

よく引き合いに出される例として，MSS が 1.5KB，RTT が 0.1 秒として，安定した輻輳回避フェーズでの平均スループット 10Gbps を実現したい場合を考える．ランダムなロス率を p とすると，平均スループットは，1.2MSS/(√p RTT) なので，$10^{10}$ = (1.2*1500*8)/(√p *0.1) より，p = $(1.2*1500*8*10^{-9})^2$. これは 4822500000 パケットに 1 個（約 100 分に 1 回）しかロスが起きてはならないことを意味し，10Gbps の平均スループットの達成は非現実的といえる.

さて，[1] は誤り制御にかかわる根本的な問題で，エンドツーエンドのプロトコルに頼る限りは，回避は難しい．そこで，cw を大きく取れるような環境を用意し，その上で，[2] や [3] を解決して cw を素早く適切に増加させたい．cw を大きく取るには，(i) 送信側ソケットバッファ（再送に備えて ACK が終わっていない送信データを保持）が大きい；(ii) 受信側ソケットバッファ（アプリケーションが引き取るまで受信データを保持）が大きい；(iii) Window Scale および Timestamp オプションを用いて，プロトコル上大きな SN や AN を利用できる；ことが必要である．なお，(iii) の問題は，すでに 1988 年の RFC1072 で必要性が指摘され，最近の OS は対応済みである.

結局，[2] や [3] が問題になる．[2] における cw の小さな初期値からの増加に時間がかかる問題には，単純には初期 cw を増やす方法がある．従来の標準は 1MSS 分であったが，RFC3390 では 4KB まで増やすことが提案され，すでに普及が始まっている．ただし，このような固定値で一律に大きくできる範囲は限られており，ごく短いファイル転送の効率は上がるが，大容量転送への効果は少ない．なお，インターネット上のさまざまな WWW サーバの cw 初期値を調べた実験結果が公開されている ☆5.

一般に，MTU（MSS）を大きくしたり，複数の TCP コネクションを用いて並列に転送したりすることで，ある程度のスループット向上が期待できる．しかし，より効果的な解決のためには，スロースタートや AIMD アルゴリズムに改良を加える，あるいはそれらを棄てて新しい輻輳制御を導入する，等の本質的な変更が避けられない．そこで，そのような長距離高速通信向けのトランスポートプロトコルの研究を次章で紹介する．なお，既存の TCP（標準 TCP）のチューニングに関しては，詳細なノウハウが公開されているので参照されたい ☆6.

一方，さまざまな理由から，エンドツーエンドプロトコルでの誤り・輻輳制御を諦め，パスの途中で TCP を何段か中継し，中継区間ごとに制御を行う方式も研究されている．オーバーレイやミドルボックスと呼ばれる形態の一種であり，RFC2757（“Long Thin Networks”）の中の Split TCP や Performance-Enhancing Proxies も含まれる．もし各中継区間の RTT が短くなるように配置できれば，この方式の導入によって，長大 RTT に起因する問題は回避できる.

## 高速 TCP に向けての研究開発動向

長距離高速通信という立場から，TCP のスロースタートや輻輳回避の制御の変更を提案している最近の研究をいくつか紹介する.

### スロースタートに関して

大きな cw に達するまでに時間がかかるという問題には，「クイックスタート」☆7 と呼ばれる方式が提案されている．これは，TCP コネクション確立時に交換するパケットのヘッダ内に新たにパラメタを定義し，途中の各ルータに現時点の可用帯域幅を書き込んでもらい，それを元に適切な（大きな）値を初期 cw として与えるものであるが，ルータの対応を前提としており，短期的に実現可能な解ではない.

一方，ネットワークに過度の輻輳を引き起こす問題には，cw がある閾値（たとえば 100 MSS）を超えたら，指数増加を止めて，しだいに増加速度を落としていく方式が提案され，「制限付きスロースタート」☆8 と呼ばれている.

これらは，以下の GridFTP, HSTCP, Scalable TCP 等と組み合わせて使うことができる.

☆5 http://www.icir.org/tbit/June2003/
☆6 http://www.psc.edu/networking/perf_tune.html や，http://www-didc.lbl.gov/TCP-tuning/
☆7 http://www.icir.org/floyd/quickstart.html
☆8 http://www.icir.org/floyd/papers/draft-floyd-tcp-slowstart-00b.txt

### GridFTP

複数の TCP コネクションを用いる並列転送によって合計スループットを向上させることができるが，GridFTP☆9 は，そのメリットを利用できるファイル転送アプリケーションであり，ファイル内を任意の部分に分割して，それらを並列に転送することで 1 つのファイルの高速転送を実現する．さらに，転送対象ファイルが分割または複製によって複数の地点に分散配置されている場合には，それらを並行して転送することで，複数のパスを利用した高速転送が可能になる．これは，RAID ディスクにおけるアクセスの高速化と類似している．今すぐに利用可能な手段として，Grid コミュニティでは期待が高い．

### HSTCP, Scalable TCP

輻輳回避に関して，AIMD アルゴリズムを改良し，cw が大きい場合にのみ，正常 ACK 時の cw の増加を速くし，かつ，間欠ロスが起きた時の cw の減少を遅くする方式として，High Speed TCP（HSTCP)☆10 や Scalable TCP☆11 が提案されている．

一般に，AIMD アルゴリズムは，正の値 A, B（B<1）を使って，正常な ACK ごとの増加を cw = cw + A，間欠ロス発生時の減少を cw = cw*(1 −B) と書ける．標準 TCP は，A = MSS/cw, B = 0.5 である．それに対して HSTCP や Scalable TCP は，cw がある閾値（たとえば 38MSS）以下であれば，標準 TCP と同じ振る舞いをするが，その閾値を超えると，

- HSTCP：A = a (cw) /cw：a ( ) は cw のある増加関数，
  B = b (cw)：b ( ) は cw のある減少関数，
- Scalable TCP：A = 0.01：定数，B = 0.125：定数．

その結果，HSTCP も Scalable TCP も，cw が大きい場合には，間欠ロスが起きても cw を素早く回復し，BDP が大きい環境においては，平均 cw を標準 TCP より大きくすることができる．たとえば，HSTCP は，前章で述べた，MSS が 1.5KB, RTT が 0.1 秒の環境での輻輳回避フェーズ内で 10Gbps の平均スループットを実現する例においては，パケットロスが 12 秒に 1 回起きてもそのスループットを達成できる．

なお，標準 TCP との共存に関して，BDP が大きい場合の HSTCP による標準 TCP への圧迫（不公平性）を改善するための研究もある．一方，Scalable TCP は HSTCP 以上にアグレッシブであり不公平性も大きい．また，性能の安定性や公平性は，パスの途中のルータにおけるバッファ長やバッファ管理方式（特に，連続ロスの発生や利用者間のバッファ占有の不公平性を緩和するために戦略的にパケットを廃棄する Active Queue Management の場合）に大きく依存すると考えられる．

### FAST TCP

標準 TCP では，ロスによってのみ輻輳を検出するが，事前に輻輳を検出してロスの発生を防ぐように送信速度を調整することが有効な場合がある．そのため，たとえば，TCP Vegas のような，RTT の増大によって輻輳を推定する方法や，RFC3168 の Explicit Congestion Notification（ECN）のような，パス上のルータから輻輳を通知してもらう方法が従来から提案されてきた．そして，それらを発展させた長距離高速通信向けの TCP として，Fast Active-queue-managed Scalable TCP（FAST TCP）が提案されている☆12．

FAST TCP は，標準 TCP の AIMD アルゴリズムとはまったく異なる，時間遅れを含む非線形フィードバック制御系の平衡問題に基づいた輻輳制御を採用している．制御系の原理に基づいた cw の制御を行うための具体的な方式として，TCP Vegas と同様に RTT を常時計測し，最小 RTT と現 RTT によって輻輳状況を推定する方式と，ECN を用いてパス上のルータの輻輳状況を知る方式とが提案されているが，実装・実験が報告されているのは前者だけのようである．この FAST TCP と HSTCP や Scalable TCP との性能比較も公開されている☆13．

### XCP, CADPC

HSTCP や FAST TCP が標準 TCP 互換（受信側が標準 TCP でも動作可能）であるのに対して，以下は TCP とはまったく別のトランスポートプロトコルであり，ルータのサポートを前提としている．これらは，一般に同種のプロトコルのみが共存する場合は，標準 TCP より，スループット，安定性（輻輳を事前に回避），公平性が優れていると期待できる．しかし，その精緻な制御は，標準 TCP のような大雑把なプロトコルのコネクションと共存すると，マイナスに働く可能性もある．

eXplicit Congestion control Protocol（XCP）☆14 は，エンドホストとルータ間で ECN よりも精密な情報の交換を行い，それに基づいて輻輳制御を行う．具体的

☆9 http://www-fp.globus.org/datagrid/gridftp.html
☆10 http://www.icir.org/floyd/hstcp.html
☆11 http://www-lce.eng.cam.ac.uk/~ctk21/scalable/
☆12 http://netlab.caltech.edu/FAST/
☆13 http://www-iepm.slac.stanford.edu/monitoring/bulk/fast/
☆14 http://www.ana.lcs.mit.edu/dina/XCP/

には，各パケット内の XCP ヘッダには，(1) 現 cw，(2) 現 RTT，(3) cw の増減量，のフィールドがあり，送信側は，自分が持っている (1) と (2) の値，自分が希望する (3) の値を入れてパケットをパスに沿って受信側へ投げる．途中の各ルータは，(1)，(2)，および出力リンクの可用帯域幅等から判断して，(3) の値が自分にとって許容できるかを判断し，許容できない場合は，書き換える．結局，受信側に届いたパケットには，パス上の全ルータが許容できる (3) の値が格納されているので，その情報を送信側に返し，送信側はそれに基づいて，実際に cw を更新する．

Congestion Avoidance with Distributed Proportional Control/Performance Transparency Protocol (CADPC/PTP) [☆15] は，PTP を用いてパスに沿った可用帯域幅を知り，その情報に基づいて，CADPC という方式で輻輳制御を行う．送信側が PTP パケットをパスに沿って受信側へ投げると，途中の各ルータは，出力リンクの (1) IP アドレス，(2) 帯域幅，(3) バイトカウンタ，(4) タイムスタンプ，を書き込むので，これを 2 回行うことで，その間のバイトカウンタの増加から可用帯域幅を知る．XCP と違って，ルータにおける複雑な計算をなくし，単に情報を書き込むだけにした点で，ルータにとっては軽量といえる．

| ニックネーム | 帯域[Mbps] | 距離[km] |
|---|---|---|
| JAPAN | 600 | 8,000 |
| EARTH$^2$ | 120 | 40,000 |
| EARTH$^4$ | 40 | 80,000 |
| MOON | 10 | 350,000 |

表 -1　JGN 長距離・大容量テストベッド

## JGN 長距離・大容量テストベッド上での実験

通信・放送機構では，Japan Gigabit Network (JGN) [☆16] 上を周回する複数のパターンの長距離 ATM-PVC を設定して実験用パス（**表 -1**）を用意し，「JGN 長距離・大容量テストベッド」[☆17] として一般利用を公募している．

筆者らは，このうちの JAPAN と呼ばれるパス（帯域幅：600Mbps，RTT：0.1 秒，BDP:7.5MB）上で iperf [☆18] コマンドを用いて TCP データ転送を行い，標準 TCP, HSTCP, Scalable TCP のスループット特性を調べた．

実験環境，機器構成を**図 -2** と**表 -2** に示す．パスに沿った最小 MTU は 1.5KB で，TCP オプションは，

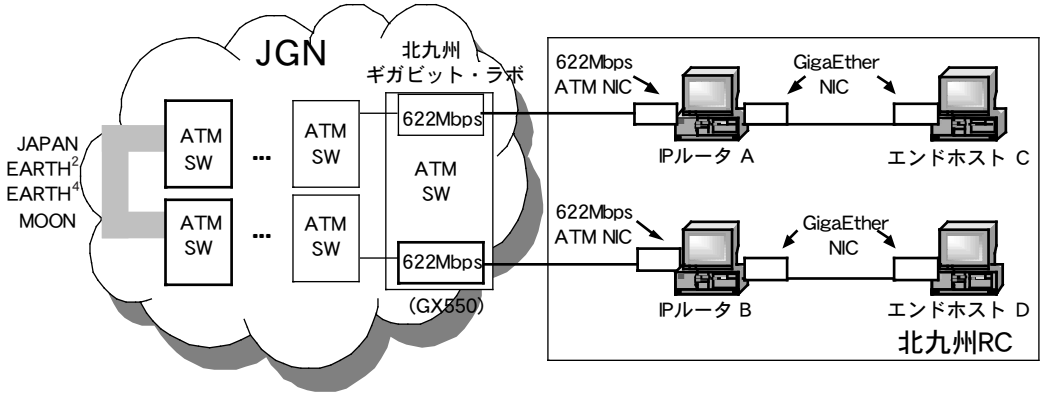


図 -2　実験ネットワーク

| | IP ルータ | エンドホスト |
|---|---|---|
| OS | Red Hat Linux 8.0 Kernel 2.4.18-26.8.0 | Red Hat Linux 9.0 Kernel 2.4.20-8 |
| CPU | PentiumIV 2.0 GHz | |
| メモリ | PC2700 512MB | PC2700 1GB |
| PCI パス | 32 ビット | |
| HD | 内蔵用可搬型 HD 60GB,7200 rpm | |
| EtherCard | Intel Pro/1000MT | |
| ATM NIC | Fore(Marconi)HE 622 SMF | - |
| 追加パッケージ | Linux - atm<br>Fore HE 用ドライバ | Iperf1.6.5 ( 計測ツール ) |

表 -2　実験機器構成

☆15 http://fullspeed.to/ptp/
☆16 http://www.jgn.tao.go.jp/
☆17 http://www.noc.jgn.tao.go.jp/longdist.htm
☆18 http://dast.nlanr.net/Projects/Iperf/

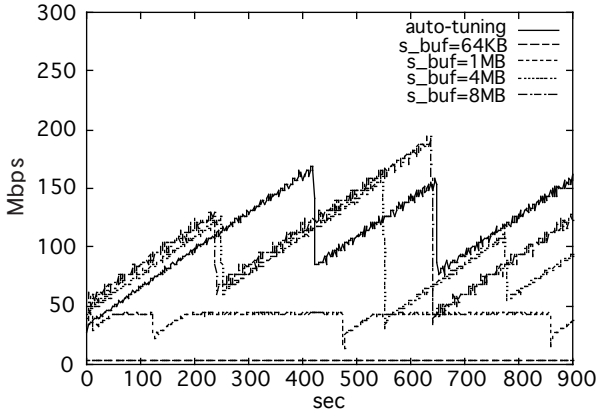


図 -3　送信バッファ長による比較

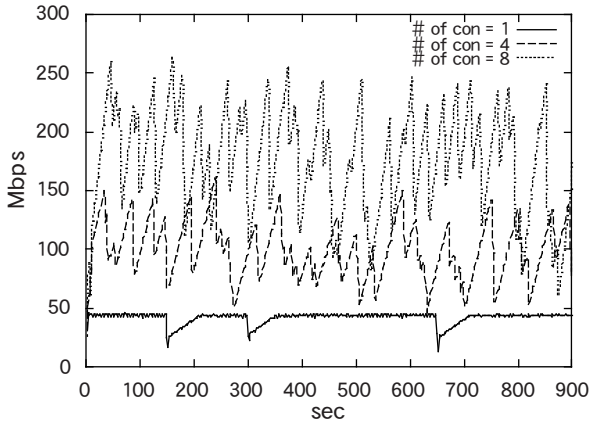


図 -4　並列コネクション数による比較

MSS 通知，SACK, Window Scale, Timestamp が有効であった．また，2 台のルータの ATM NIC 間を光ケーブルで直結した環境では，標準 TCP を用いて 1 秒平均のスループットが安定して 314Mbps であった．

まず，標準 TCP を用いて送信（ソケット）バッファ長の効果を確認したのが**図 -3** である．受信（ソケット）バッファ長は 16MB と十二分に大きく取り，送信バッファ長の設定を，64KB から 8MB, および自動チューニング[☆19] に変化させて，2 秒平均のスループットの推移を見た．送信バッファ長が 1MB ではまだ頭打ちの要因になっていること，自動チューニングはうまく動作していること，900 秒間に渡って輻輳回避フェーズで安定しているが間欠ロスが発生していることが分かる．

標準 TCP を用いて並列コネクション転送の効果を確認したのが**図 -4** である．各コネクションの送信バッファは 1MB で，1,4,8 本の場合の 2 秒平均の合計スループットの推移を見た．合計スループットの増加速度（傾き）はコネクション数に比例している様子が分かる．また，図 -3 と比較すると，8MB 送信バッファのコネクション 1 本と 1MB 送信バッファのコネクション 8 本の合計では，後者の平均スループットの方がかなり高く，並列転送の有効性を示している．

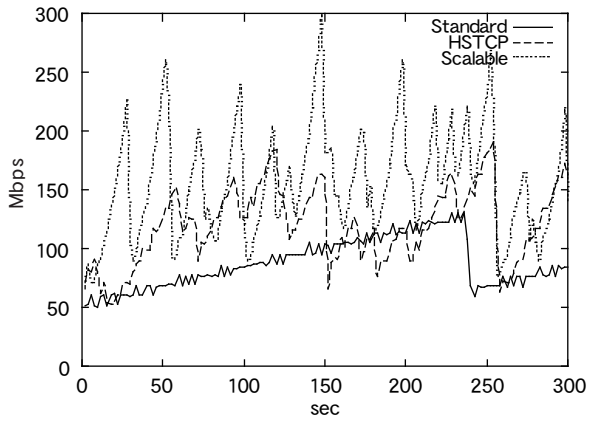


図 -5　標準 TCP, HSTCP，Scalable TCP の比較

標準 TCP，HSTCP（G. Fairey の実装）[☆20]，Scalable TCP（T.Kelly の実装）[☆11] の特性比較を行ったのが**図 -5** である．送信バッファ長は 8MB と十分大きく取って，各 TCP1 コネクションでの 2 秒平均のスループットの推移を見た．標準 <HSTCP<Scalable TCP の順でより激しい変動を示し，300 秒平均のスループットはその順に増加した（3 回の実験の平均で，70<105<142 Mbps）が，同時に，パケットロス率もその順で増加した（3 回の実験の平均で，0.017<0.027<0.054 %）．

## おわりに

ここまで述べてきたように，長距離広帯域パス上で効率的な通信を行うことは簡単ではなく，今後，それを利用するアプリケーションも含めたさまざまな工夫が必要になる．上記の通信・放送機構の JGN 長距離・大容量テストベッドは，一般研究者が実証的研究を行うための環境を提供している．また，そのテストベッドを用いた研究のコンテストが平成 15 年 5 月 30 日から実施されており[☆17]，特に優れた研究は表彰されることになっている．ネットワーク研究者・アプリケーション研究者の方は，一度挑戦されてはいかがでしょうか？

**参考文献**

1）村山公保，西田佳史，尾家祐二：岩波講座インターネット第 3 巻，トランスポートプロトコル，岩波書店（2001）．

（平成 15 年 8 月 5 日受付）

☆19 http://www.csm.ornl.gov/~dunigan/net100/auto.html
☆20 http://www.hep.man.ac.uk/u/garethf/hstcp/