コネクション分割によるTCPスループット向上システムの仮想化環境における評価

Evaluation on Virtual Machine Environment of
TCP Accelaration System by Dividing TCP Connection

升屋　正人*, 室屋　孝英†, 下園　幸一*
Masato MASUYA*, Takahide MUROYA†, and Koichi SHIMOZONO*

鹿児島大学 学術情報基盤センター*
鹿児島大学 大学院理工学研究科 情報生体システム工学専攻†
Computing and Communications Center, Kagoshima University*
Department of Information Science and Biomedical Engineering,
Guraduate School of Science and Engineering, Kagoshima University†

通信回線の広帯域化に伴い，大容量の通信が可能となった．しかし，長距離通信において帯域を十分に使い切れないという問題が顕在化している．これは，TCPスループットが往復遅延時間の影響を受けるためである．そこで本研究では，プロキシを用いてコネクションを分割するTCPスループット向上システムを仮想化環境内に構築し，このシステムの有効性を検証した．

キーワード ：LFN(Long Fat Network)，高帯域遅延積ネットワーク，ブロードバンド，高遅延インターネット

Transmission of large amounts of data became available with increasing bandwidth of access lines. However, TCP underperforms when sending bulk data across high-speed links. It is a well-known characteristic of TCP that as the round trip time grows, end-to-end throughput decreases. In order to accelerate TCP throughput on long-haul transfer, we developed novel system by dividing TCP connection and evaluated that system on virtual machine environment.

Keywords : LFN(Long Fat Network), high bandwidth delay products network, broadband, long-delay Internet

1. はじめに

伝送路の帯域が数十から数百 Mbps を超えるブロードバンドアクセス回線の普及により，LFN（Long Fat Network）や高帯域遅延積ネットワークとよばれる，帯域幅が大きく遅延が大きいネットワークにおいて，十分なTCP（Transmission Control Protocol）スループットを得ることができない問題が顕在化しつつある．

大容量のデータを遠隔地間で送受信するグリッドコンピューティング，様々な組織のサーバが集約する東京から離れた地域におけるインターネットアクセス，遅延が大きい衛星通信及び移動体通信，災害に備えた遠隔地のデータセンターへのデータレプリケーションなど，高速大容量の通信が必要であるにも関わらず十分な帯域が得られていない様々な分野でTCPスループットの向上が必要となってきている．

インターネットで広く使われている通信方式

*〒 890-0065　鹿児島市郡元 1–21–35
1–21–35, Korimoto, Kagoshima 890-0065
E-mail: {masatom, simozono}@cc.kagoshima-u.ac.jp
†〒 890-0065　鹿児島市郡元 1–21–40
1–21–40, Korimoto, Kagoshima 890–0065
E-mail: muroya@bio.cc.kagoshima-u.ac.jp

であるTCP[1]ではデータは分割して送信されており，一定量の分割データごとに到達確認を行うことで通信の確実性が保証されている．この一定量（TCPウィンドウサイズ）の最大値は，TCPヘッダのウィンドウサイズフィールドが16ビットであることから，最大$2^{16}-1=65,535$バイト，およそ64Kバイトとなる．確認応答の受信後に次のデータが送信されるため，データを送信し確認応答が帰ってくるまでの往復遅延時間をt秒とすると，t秒間に送信できる最大のデータ量が64Kバイトということになる．このときの最大のTCPスループットは，$64\times 8\div t$ Kbpsとなる（1バイト＝8ビット）．例えばtが0.05秒（＝50ミリ秒）のとき，数百Mbpsを超える帯域幅があったとしても10 Mbpsが理論的な最大のTCPスループットとなる．これがLFNにおいて十分なTCPスループットを得られない原因である．

実際のTCP通信においては，送信するデータ量を次第に増やすスロースタートと呼ばれる動作や，パケット損失が発生した場合に送信データ量を調整する輻輳制御の仕組みがある[2]．また，システムによってはTCPウィンドウサイズを最大1Gバイトまで拡大するウィンドウスケールオプション[3]が実装されている場合もある．このため，単純に往復遅延時間から最大TCPスループットを求めることはできない．近年のシステムにおける最大TCPスループットは，TCPウィンドウサイズと往復遅延時間のみを考慮して求めた前述の値より大きくなるのが一般的である．しかし，往復遅延時間以外の条件が同じであれば，往復遅延時間が大きければ大きいほど，TCPスループットが低下することは明らかである．

逆に言えば，往復遅延時間を小さくすればTCPスループットが向上することになる．TCP通信を経路を変えずにエンドツーエンドで行う限り往復遅延時間は短縮できないが，経路の中間に設置したプロキシでTCPコネクションを複数に分割すればTCPコネクションごとの往復遅延時間を小さくできる．これによりTCPスループットを向上できるはずである．

われわれは，このことに着目し，TCPコネクションを複数に分割することで，LFNにおけるTCPスループットを向上させることにした．東京から離れた地域からのインターネットアクセスが遅いなど，高遅延インターネットにおいて十分なTCPスループットを得られない問題は，本研究の方法を実装したシステムを実際のインターネット環境に導入することにより解決できる．

LFNにおけるTCPスループット向上に関する研究は多く行われている．しかし，そのほとんどがTCPプロトコルに手を加えてエンドツーエンドのTCPスループットの向上を図るものである．TCPコネクションの分割の例としては，光ファイバ回線に衛星回線や無線回線が接続する場合など，帯域や信頼性が異なる回線が混在した伝送路を経由した通信においてその中間にプロキシを置く方法[4),5)]が提案されている．しかし，同等品質の伝送路上でTCPコネクションを分割することによりTCPスループットの向上を図る例は少ない．牧ら[6]はコネクション分割によるTCPスループット向上を提案しているものの，モデルの検討に止まる．小林と中山[7]はTCPコネクション分割の実環境での評価までを行っているが，経路外の中継ノードを利用する方法の提案であり，2分割までしか検討の対象としていない．山根木ら[8]のものは両端にプロキシを置く方法であり，コネクション分割とは言えない．TCPコネクションを分割することで高遅延インターネットにおけるTCPスループットを向上させるシステムはこれまで存在していない．

そこで本研究では，TCPコネクションを複数に分割することによりTCPスループットの向上を図ることができるのか，また，どの程度向上させることができるのかを，仮想化環境で調べることにした．

2. TCPコネクションの分割

高遅延ネットワークにおけるTCPスループットの向上に関する取り組みはこれまでにも行われている．そのほとんどはTCPプロトコルそのものに手を加える方法である．前述のウィンドウスケールオプションのほか，初期ウィンドウサイズを大きくする方法[9]や輻輳制御アルゴリズムの改良などが行われてきた．最近では複数のTCPセッションを並列で用いる方法や新たなプロトコルを用いる方法も検討されてい

る．これらを実装した新しいシステムを送信側と受信側，双方に導入することで，TCP スループットの向上が可能になる．しかし，これらの方法では既存のシステムの TCP スループットは向上できない．既存のシステムに手を加えずに TCP スループットを向上させるため，PEP（Performance Enhancing Proxy）を両端や中間に設置する方式[4] を用いることになる．

本研究では，経路の中間に設置したプロキシで TCP コネクションを複数に分割し，それぞれの区間で往復遅延時間を小さくすることで TCP スループットの向上を図る（図 1）．PEP で用いられている方法とは異なり，用いるプロキシ自体には TCP スループット向上の仕組みは実装しない．これによりプロキシを用いて往復遅延時間を短縮するだけで TCP スループットの向上が可能であるのか，また，どの程度向上できるのかを調べることができる．

図 1　TCP コネクション分割の概念図

2.1　分割数と TCP スループットの関係

パケットロスがなく輻輳が発生しない理想的なネットワークでは，受信ウィンドウサイズを $rwin$，クライアントとサーバの間の往復遅延時間を RTT として，TCP スループットの最大値 T を以下の式で表すことができる．

$$T = \frac{rwin}{RTT} \tag{1}$$

伝送路を n 区間に分割した時の区間 1 から区間 n までの往復遅延時間をそれぞれ，RTT_1, RTT_2, ..., RTT_n とし，$n-1$ 台設置するプロキシによるオーバーヘッドを $f(n-1)$ とすると，往復遅延時間が最大の区間がボトルネックとなる．最大の往復遅延時間が RTT_{max} のとき，区間を n 分割したときの最大の TCP スループット T_n は，

$$T_n = \frac{rwin}{RTT_{max}} - f(n-1) \tag{2}$$

となる．このとき，

$$RTT_{max} = \max(RTT_1, RTT_2, ..., RTT_n) \tag{3}$$

また，

$$RTT = RTT_1 + RTT_2 + ... + RTT_n \tag{4}$$

である．

T_n が最大となるのは往復遅延時間がすべて等しいとき，すなわち，

$$RTT_1 = RTT_2 = ... = RTT_n = \frac{RTT}{n} \tag{5}$$

となるように区間を分割した時で，このとき，

$$T_n = T \times n - f(n-1) \tag{6}$$

となる．分割しない場合の TCP スループット T を n 倍した値からオーバーヘッドを除いたものが n 分割した場合の TCP スループット T_n となる．

2.2　プロキシシステム

本研究の TCP スループット向上システムは，経路上に設置する複数のプロキシから構成される．実際のインターネット環境への導入を想定し，それぞれのプロキシを独立したサーバ上に構築することにした．

2.2.1　オペレーティングシステム

経路上に設置するプロキシのオペレーティングシステムとしては，サーバ用途に広く用いられている Linux を使用する．Linux には様々なディストリビューションがあるが，入手が容易で利用者も多いことが想定され，技術情報も豊富にあることから，CentOS[10] 6.4 を用いることにした．Linux カーネルバージョンは 2.6.32 である．

Linux 2.6.32 では複数の輻輳制御アルゴリズムが使用できるが，本研究では Linux 2.6.19 以降で標準となっている CUBIC TCP[11] を用いる．また，カーネルパラメータの変更によるチューニングは行わず，すべてインストール時のデフォルト値を使用する．インストールは最小限インストールで行い，使用するプロキシソフトウェアのみを後から追加することにした．

2.2.2 TCPプロキシソフトウェア

各プロキシにはTCPプロキシ機能を実現するためのソフトウェアを導入する．TCPプロキシとして利用できる仕組みはいくつか存在しているが，本研究では任意のTCPプロトコルを中継することができるSOCKS[12]を用いる．SOCKSプロトコルに対応したプロキシソフトウェアとして，Dante[13]を使用することにした．DanteはSOCKSサーバ，SOCKSクライアントの双方に対応しており，多段プロキシを実現できる．

CentOSへのDanteの導入はRepoForgeリポジトリ[15]から行う．使用したバージョンはdante 1.3.2である．

また，SOCKS以外のTCPプロキシについてもTCPコネクション分割での利用可能性及び性能を確認するため，HTTPプロキシおよびFTPプロキシとして利用することができるSquid[14]についても併せて評価することにした．

CentOSへのSquidの導入はCentOSベースリポジトリから行う．使用したバージョンはSquid-3.1.10である．Squidにはコンテンツキャッシュ機能があるが，本研究ではプロキシとしての評価を行うため，コンテンツキャッシュ機能を無効にして測定を行った．

2.2.3 TCPプロキシの構成

プロキシは図1のように，数珠つなぎで連結する．

経路を2分割する場合，経路上に1台設置されるプロキシP1上でプロキシソフトウェアを動作させ，クライアントCは，プロキシP1を経由してサーバSとTCP通信を行う．クライアントCとサーバSの間の経路がC–P1とP1–Sの2区間に分割されることになる．

経路を3分割する場合，クライアントCは経路上に2台設置されるプロキシP1とP2のうち，経路上近い位置にあり往復遅延時間の短いプロキシP1をプロキシとして指定してサーバSとTCP通信を行う．プロキシP1はプロキシP2をプロキシとしてサーバSと通信を行う設定とする．これにより，クライアントCはプロキシP1とプロキシP2の両方を経由してサーバSと通信を行うことになる．クライアントCとサーバSの間の経路が，C–P1，P1–P2，P2–Sの3区間に分割されることになる．

経路を4分割する場合は3台のプロキシ，5分割する場合は4台のプロキシを用いて同様の構成とする．例えば5分割の場合は，C–P1–P2–P3–P4–Sと連結することになる．本研究では5分割までの検証を行うことにした．

3. TCPスループットの測定

TCPスループットは，HTTPプロトコルまたはFTPプロトコルで100 MBのファイルをダウンロードし，要した時間を測定することにより行う．100 MBのファイルをダウンロードするのに要した時間をs秒とすると，TCPスループットT_nは以下の式で表される．

$$T_n = \frac{100}{s}(\mathrm{MB/s}) \tag{7}$$

ダウンロードに使用する100 MBのファイルは，ddコマンドを用い，

```
# dd if=/dev/urandom of=100m bs=1M count=100
```

として生成した．HTTPサーバ，FTPサーバとも圧縮を行う設定は行わないが，ファイルの中身を乱数とすることで圧縮の影響を排除できる．

サーバS上で動作させるHTTPサーバソフトウェアにはCentOSベースリポジトリからインストールできるApache httpd 2.2.15を使用した．FTPサーバソフトウェアにはCentOSベースリポジトリからインストールできる，vsftpd-2.2.2を使用した．

TCPスループットの測定を行うクライアントCでは，HTTPとFTPの双方に対応し，プロキシホストの設定も可能なファイル転送ツールであるcurl 7.19.7をCentOSベースリポジトリからインストールし，HTTPプロトコルおよびFTPプロトコルでのダウンロード及び時間の測定に使用した．

クライアントC，サーバS，プロキシP1，P2，P3，P4はすべて仮想化環境内で仮想マシンとして用意した．

3.1 仮想化環境

仮想化環境は KVM（Kernel-based Virtual Machine）[16] を用いて構築した．KVM は Linux カーネルに仮想マシンの管理機能を統合したもので，完全仮想化による仮想マシン環境を実現できる．CPU の仮想化拡張機能 (Intel VT または AMD-V) を利用しているため，仮想化に対応した CPU が必要である．KVM は Linux 2.6.20 からカーネル標準機能となっている．表 1 に仮想化環境に用いたサーバの仕様を示す．

表 1　仮想化サーバホストの仕様

OS	CentOS 6.4 x86_64 Linux 2.6.32
機器名	富士通 PRIMERGY TX200S5
CPU	Intel Xeon X5570(2.93 GHz) × 2 （計 8 コア 16 スレッド）
メモリ	12 GB
HDD	SAS 450 GB × 4 (RAID6)

仮想マシン 6 台はすべて表 2 のように作成した．

表 2　仮想マシンゲストの仕様

OS	CentOS 6.4 x86_64 Linux 2.6.32
CPU	1
メモリ	1GB
HDD	8GB
NIC	virtio

すべての仮想マシンの `eth1` をホスト側で用意した仮想ブリッジ `virbr1` に接続し，同じネットワークアドレスの独立の IP アドレスを割り当て，分割数に応じて netem を用いて遅延をそれぞれの仮想マシンの `eth1` に設定した．パケットが `eth1` を出る時に netem で設定した遅延がかかることになる．仮想 NIC として用いた virtio を介した仮想マシン間の TCP スループットは 400 MB/s（3.2 Gbps）以上であることを確認しており，本研究における TCP スループットの測定に影響しない．

ホストマシンのコア数とメモリ量は，仮想マシンで使用する CPU 数およびメモリ量を上回っており，各仮想マシンの動作に他の仮想マシンの動作の影響はない．ホストマシンにおける仮想化環境の管理についても別のホストから行うこととし，仮想化環境への影響が無いようにした．

3.2 往復遅延時間の設定

遅延の設定は以下のコマンドで行った（10 ミリ秒の遅延を eth1 に設定する例）．

```
# tc qdisc add dev eth1 root netem delay 10ms
```

プロキシを入れない場合も含め，クライアント `C` とサーバ `S` の間の往復遅延時間が合計で 50 ミリ秒になるように設定した．この往復遅延時間は国内のインターネットにおける東京との往復遅延時間の最大値に近い値である．

3.2.1 2 分割の場合の遅延の設定

TCP コネクションを 2 分割した場合の TCP スループットは，2 つの区間の往復遅延時間がそれぞれどのような値になるかで異なることが想定される．経路上でどの位置にプロキシを設置した場合に最も高い TCP スループットとなるのかを検証するため，それぞれの仮想マシンの `eth1` に表 3 の通り遅延を設定した．

表 3　2 分割の場合の遅延の設定（単位：ms）

C	P1	S
40	0	10
30	0	20
25	0	25
20	0	30
10	0	40

プロキシ `P1` には遅延の設定を行っていない．クライアント `C` に設定した遅延がクライアント `C` とプロキシ `P1` の間の往復遅延時間，サーバ `S` に設定した遅延がプロキシ `P1` とサーバ `S` の間の往復遅延時間になる．

3.2.2　3 分割以上の場合の遅延の設定

TCP コネクションを 3 分割以上とする場合，合計の往復遅延時間を 50 ミリ秒とし，すべての区間の往復遅延時間が等しくなるように設定した．

3 分割の時はクライアント C，プロキシ P1，P2，サーバ S のすべてに 8.33 ミリ秒の遅延を設定した．これによりすべての区間の往復遅延時間が 16.66 ミリ秒となる．

4 分割の時はクライアント C，プロキシ P1，P2，P3，サーバ S のすべてに 6.25 ミリ秒の遅延を設定した．これによりすべての区間の往復遅延時間が 12.5 ミリ秒となる．

5 分割の時はクライアント C，プロキシ P1，P2，P3，P4，サーバ S のすべてに 5 ミリ秒の遅延を設定した．これによりすべての区間の往復遅延時間が 10 ミリ秒となる．

4.　測定結果と考察

3.2.1，3.2.2 に示した各遅延設定の条件下において 100 MB のファイルをダウンロードし，TCP スループットを測定した．測定はそれぞれ 10 回行っており，以下に示す値はすべて中央値である．

4.1　分割位置と TCP スループット

コネクション分割の位置と TCP スループット向上の効果の関係を明らかにするため，TCP コネクションを 2 分割し，3.2.1 の通りに遅延を設定して TCP スループットを測定した．

表 4　2 分割の場合の FTP スループット (プロキシは dante)

遅延 (C–S) (ms)	スループット (MB/s)	向上率 (%)
0	10.4	100
40–10	12.4	119
30–20	16.3	157
25–25	17.7	170
20–30	16.4	158
10–40	12.8	123

TCP プロキシソフトウェアとして dante を用い，クライアント C とプロキシ P1，プロキシ P1 とサーバ S，それぞれの間の往復遅延時間を変えて，FTP サーバから 100 MB のファイルをダウンロードして TCP スループットを測定した．結果を表 4 に示す．

すべての場合でコネクション分割により TCP スループットは向上し，各区間の往復遅延時間を同じに設定した時の TCP スループットが最大になった．

また，TCP プロキシソフトウェアとして squid を用い，dante を用いた FTP サーバからのダウンロードと同様に各区間の往復遅延時間を変えて，HTTP サーバから 100MB のファイルをダウンロードして TCP スループットを測定した．結果を表 5 に示す．

向上率は異なるものの，dante を用いた FTP スループットの測定結果と同様，すべての場合でコネクション分割により TCP スループットが向上し，各区間の往復遅延時間を同じに設定した時の TCP スループットが最大になった．

式 (1) に示した通り，往復遅延時間以外の条件が同じであれば，TCP スループットは往復遅延時間が大きければ大きいほど小さくなる．このため，往復遅延時間が異なる区間ではそれぞれ TCP スループットが異なる．また，複数の区間を経由する通信においては，最もスループットが低い区間がボトルネックとなるため，式 (2) に示した通り，最も往復遅延時間の大きな区間の TCP スループットが全体の TCP スループットとなる．表 4，5 の結果とも，この傾向にあり，2.1 での理論的検討が正しいことを示した．また，TCP スループットが最大と

表 5　2 分割の場合の HTTP スループット (プロキシは squid)

遅延 (C–S) (ms)	スループット (MB/s)	向上率 (%)
0	11.0	100
40–10	15.0	136
30–20	19.5	177
25–25	20.8	189
20–30	18.1	165
10–40	13.6	124

なるのは，式 (5) に示した通り，それぞれの区間の往復遅延時間が同じ時で，区間を等分割した場合である．測定結果からもこのことが裏付けられた．

4.2 分割数と TCP スループット

TCP コネクションを 3 分割以上とした場合に，どの程度 TCP スループットが向上するのかについて，分割数を増やして測定した．以下のコネクション分割においては 3.2.2 で述べた通り，すべての区間の往復遅延時間を同じに設定している．

TCP プロキシソフトウェアとして dante を用い，分割数を 1 から 5 まで変えて FTP スループットを測定した結果を表 6 に示す．分割数 1 は分割しないことを示しており，プロキシを経由しない．

表 6 分割数と FTP スループット (プロキシは dante)

分割数	区間の遅延 (ms)	スループット (MB/s)	向上率 (%)
1	50	10.4	100
2	25	18.4	177
3	16.7	26.1	251
4	12.5	32.4	312
5	10	40.1	386

すべての場合で TCP スループットは向上している．表 6 より，分割数 n と TCP スループット T_n の関係は，分割しない場合の TCP スループットを T として以下の式で近似的に表すことができる．

$$T_n = T \times (0.3 + 0.7n) \tag{8}$$

この式は区間数が 1 区間増えるごとに，TCP スループットが 70%向上することを示している．TCP スループットは理論的には式 (6) で示した通り区間数倍の項とオーバーヘッドの項に分けて考えることができる．そこで，式 (8) を以下のとおり変形する．

$$T_n = T \times n - T \times 0.3(n-1) \tag{9}$$

これにより，オーバーヘッドをプロキシの数 $n-1$ の関数で表現することができ，

$$f(n-1) = T \times 0.3(n-1) \tag{10}$$

となる．プロキシ 1 台につき，区間を分割しない場合のスループットの 30%に相当するオーバーヘッドがあることになる．

なお，このオーバーヘッドは測定を行った範囲である，dante を用いた 5 分割までの区間分割において成立する．さらに分割数を増やした場合はよりオーバーヘッドが大きくなる可能性がある．

4.2.1 プロトコル及びプロキシソフトウェアによる違い

プロトコルによる違いを調べるため，TCP プロキシソフトウェアとして同じく dante を用い，分割数を 1 から 5 まで変えて HTTP スループットを測定した．結果を表 7 に示す．

表 7 分割数と HTTP スループット (プロキシは dante)

分割数	区間の遅延 (ms)	スループット (MB/s)	向上率 (%)
1	50	11.0	100
2	25	17.4	158
3	16.7	29.6	269
4	12.5	37.4	340
5	10	46.1	419

FTP プロトコルに比べ，HTTP プロトコルの方が TCP スループット，向上率とも高い値となっている．これはそれぞれに用いたサーバソフトウェアである，vsftpd-2.2.2 と Apache httpd 2.2.15 の性能の違いを反映したものと考えられる．

また，TCP プロキシソフトウェアとして squid を用い，分割数を 1 から 5 まで変えて HTTP スループットを測定した．結果を表 8 に示す．

分割数を増やすに従い，TCP スループットも向上しているが，プロキシに dante を用いた場合に比べて，4 分割から向上の度合いが小さくなり，5 分割では 4 分割と同程度の TCP スループットの向上に止まる．

表 8 分割数と HTTP スループット (プロキシは squid)

分割数	区間の遅延 (ms)	スループット (MB/s)	向上率 (%)
1	50	11.0	100
2	25	20.8	189
3	16.7	28.4	258
4	12.5	34.0	309
5	10	34.6	315

表6，7，8をグラフにしたものを図2に示す．

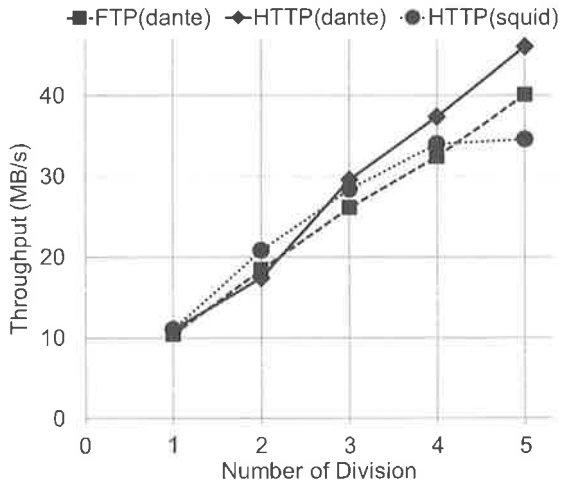

図 2　分割数と TCP スループット

プロキシに dante を用いた場合，FTP プロトコルに比べて HTTP プロトコルの方が TCP スループットの向上率が高いが，ほぼ同等の TCP スループット向上となった．一方，プロキシに squid を用いた場合，4 分割までであれば dante と同等の TCP スループットの向上であったが，5 分割では 4 分割に対する性能の向上が小さい．3 分割に対する 4 分割時の性能向上も dante に比べて低くなっている．プロキシとして squid を用いる場合，分割数が 4 分割を超えるとオーバーヘッドが大きくなる結果が得られた．このため、4 分割までであれば dante, squid のいずれを用いてもよいが，5 分割であれば dante を用いた方がよいと言える．

いずれのプロキシソフトウェアを用いた場合も，4 分割まではオーバーヘッドの影響は同等で 3 倍の TCP スループットの向上となった．

5.　まとめ

仮想化環境において，TCP コネクション分割による TCP スループット向上システムを構築し，プロキシソフトウェアとして dante と squid を用いた場合の TCP スループットを測定した．これにより，TCP コネクションを分割することで，TCP スループットを向上できることを示した．

TCP コネクション分割にあたっては，各区間の往復遅延時間が同じになるように，均等に分割した場合に最も TCP スループットの向上効果が大きい．また，分割数に応じて TCP スループットは向上し，SOCKS プロキシである dante を用いた場合，プロキシ 1 台あたり 70%の FTP スループット向上効果が得られることがわかった．

また，プロキシソフトウェアとして dante を用いる場合，HTTP スループットの向上効果は FTP より高い．一方，プロキシソフトウェアとして squid を用いた場合，4 分割までは dante と同等の HTTP スループットの向上が見られるが，5 分割では効果が低くなることが明らかになった．

本研究では往復遅延時間の短縮の効果についてのみ検証を行ったが，各プロキシで用いる輻輳制御アルゴリズムを変更するなど，PEP で用いられている技術を組み合わせることで，さらに TCP スループットを向上できると思われる．

なお，本研究の検証は他の通信の影響がない条件下で行われている．輻輳が発生する実際のインターネットに本研究のシステムを導入した場合，必ずしも同等の TCP スループット向上効果が得られない可能性がある．このため，実際のインターネットへの導入にあたっては，実サーバを用いた検証を行うとともに，輻輳の影響について調べる必要があると考えている．

謝辞

本研究の一部は，総務省戦略的情報通信研究開発推進事業（SCOPE）の委託研究「高遅延インターネットにおける TCP スループット向上システムの研究開発 (132310006)」により行われた．

参考文献

(1) Postel, J.: Transmission Control Protocol, RFC 793 (1981)

(2) Jacobson, V. and Karels, M. J.: Congestion Avoidance and Control, In SIGCOMM '88 Symposium proceedings on Communications architectures and protocols, pp. 314–329 (1988)

(3) Jacobson, V., Braden, B. and Borman, D.: TCP Extensions for High Performace,RFC 1323 (1992)

(4) Border, J., Kojo, M., Griner, J., Montenegro, G. and Shelby, Z.: Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations, RFC 3135 (2001)

(5) Ajay Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts," in Proceeding of the 15th International Conference on Distributed Computing Systems, pp. 136–143 (1995)

(6) 牧一之進, 長谷川剛, 村田正幸, 村瀬勉: TCP オーバレイネットワークにおける TCP コネクション分割機構の性能解析, 電子情報通信学会技術研究報告. IN, 情報ネットワーク, Vol. 103, No. 651, pp. 7-12 (2004)

(7) 小林弘和, 中山雅哉: 仮想リンクを利用した送受信ノード間の RTT 分割による転送スループットの向上, インターネットコンファレンス論文集 2006, pp. 105–112 (2006)

(8) 山根木果奈, 浜崇之, 長谷川剛, 村田正幸, 下西英之, 村瀬勉: TCP プロキシ機構の実ネットワーク上での実装評価, 電子情報通信学会技術研究報告. IN, 情報ネットワーク, Vol. 104, No. 658, pp. 7–12 (2005)

(9) Allman, M. , Floyd, S. and Partridge, C.: Increasing TCP's Initial Window, RFC 3390 (2002)

(10) CentOS Project: http://www.centos.org/

(11) Ha, S., Rhee, I, and Xu, L.: CUBIC: A New TCP-Friendly High-Speed TCP Variant, ACM SIGOPS Operating System Review, Vol. 42, Issue 5, pp. 64–74 (2008)

(12) Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D., and Jones, L.: SOCKS Protocol Version 5, RFC 1928 (1996)

(13) Dante - A free SOCKS server: http://www.inet.no/dante/

(14) Squid: Optimising Web Delivery: http://www.squid-cache.org/

(15) Repoforge Project: http://repoforge.org/

(16) KVM: http://www.linux-kvm.org/

(2013 年 12 月 19 日原稿受付)
(2014 年 05 月 30 日採録決定)

著者略歴

升屋　正人　1991 年東京大学理学部卒業, 1996 年同大学院農学生命科学研究科博士課程修了, 同年 4 月岡崎国立共同研究機構分子科学研究所非常勤研究員, 1997 年 11 月鹿児島大学工学部情報工学科助手, 2000 年 4 月同大学総合情報処理センター助教授, 2003 年 4 月同大学学術情報基盤センター助教授, 2006 年 11 月同教授. 博士（農学）.

室屋　孝英　2013 年鹿児島大学工学部情報工学科卒業, 鹿児島大学大学院理工学研究科情報生体システム工学専攻博士前期課程在学中. 学士（工学）.

下園　幸一　1991 年九州大学工学部情報工学科卒業, 1993 年同大学院工学研究科情報工学専攻修了, 同年 4 月九州大学情報処理教育センター助手, 1998 年 2 月鹿児島大学法文学部経済情報学科講師, 2000 年 4 月同助教授, 2007 年 7 月同大学学術情報基盤センター准教授. 修士（工学）.