Vol.2015-IOT-31 No.16
Vol.2015-SPT-15 No.16
2015/9/26

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構築と評価

升屋 正人[1,a)]　下園 幸一[1,b)]

概要：光ファイバ網の整備により通信回線は大容量化・広帯域化してきた．ところがこれに伴い，長距離通信において帯域を十分に使い切れない問題が顕在化しつつある．これは，TCP スループットが送信者と受信者の間の距離，すなわち，往復遅延時間の影響を受けるためである．距離が長くなればなるほど通信帯域は低下する．長距離通信における TCP スループットの向上を図るため，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構築し，そ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，FTTH，高遅延インターネット

Development and Evaluation of the TCP Acceleration System by Dividing TCP Connection

Masuya Masato[1,a)]　Shimozono Koichi[1,b)]

Abstract: 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 the system.

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

1. はじめに

光ファイバ網の整備が進展し，数十から数百 Mbps を超える伝送路の帯域を持つブロードバンドアクセス回線の利用が一般的となってきた．これに伴い，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も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．

TCP[1] では，データはパケットに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．この一定量である 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，$64\times8\div t$ 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．

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

[1] 鹿児島大学 学術情報基盤センター
鹿児島市郡元 1–21–35, 890–0065
[a)] masatom@cc.kagoshima-u.ac.jp
[b)] simozono@cc.kagoshima-u.ac.jp

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，オペレーティング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ず，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．経路を変えずにエンドツーエンドで TCP 通信を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．そこでまず，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について，仮想化環境で調べ有効性を検証した [4]．本論文では，仮想化環境に加え，遅延シミュレータ環境と実インターネット環境において，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有効性を評価した結果を報告する．

2. TCP コネクションの分割

LFN 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いくつか行われている．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 [3] のほか，初期ウィンドウサイズを大きくする方法 [5] 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，Google による QUIC など，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の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には，PEP（Performance Enhancing Proxy）を両端や中間に設置する方式 [6] 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．ただし，プロキシ自体には TCP スループット向上の仕組みは実装せず，コネクション分割の効果のみを評価する．これにより，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．

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．

2.2 TCP スループット向上システム

本研究の TCP スループット向上システムは，クライアントとサーバの間の経路上に設置する 1 台もしくは複数のプロキシから構成される．

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 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[7] 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．これはコネクション分割の効果のみを検証するためである．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 プロトコル [8] に対応した TCP プロキシソフトウェアである dante と，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 の２つを用いて評価を行うことにした．いずでの TCP プロキシソフトウェアも多段プロキシを実現できる．なお 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は用いていない．

3. 仮想化環境における評価

本研究ではまず，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について，仮想化環境において評価した [4].

3.1 分割位置と TCP スループット

TCP プロキシソフトウェアとして dante を用い，測定クライアントとプロキシ，プロキシとサーバ，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 (表 1).

表 1 往復遅延時間 50 ミリ秒の区間を２分割した場合の FTP スループット (プロキシは dante)．遅延は「クライアント–プロキシの遅延」–「プロキシ–サーバ間の遅延」で表している．

遅延 (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

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．これは，2.1 で示した式と一致する．

3.2 コネクション分割数と TCP スループット

TCP プロキシソフトウェアとして dante と squid を用いた FTP スループットの測定と，squid を用いた HTTP スループットの測定を，分割数を１から５まで変えて行った．サーバ–クライアント間の合計の往復遅延時間は 50 ミリ秒とした．分割数１は分割しないことを示しており，このときはプロキシを経由しない．サイズが 100 MB のファイルをダウンロードするのに要した時間を測定して TCP スループットを求めた (図 1).

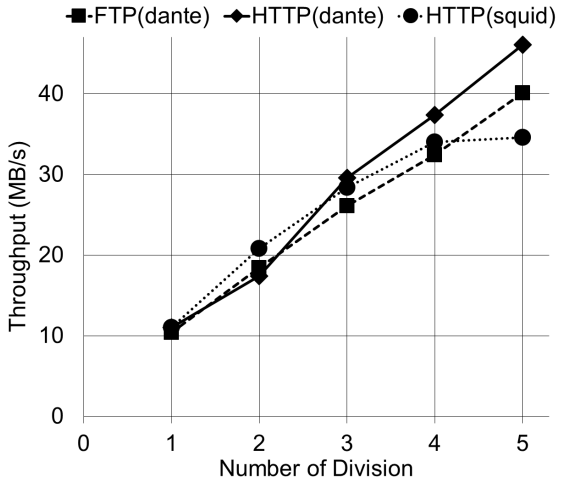

図 1 仮想化環境におけるコネクション分割数と TCP スループット (MB/s)．往復遅延時間 50 ミリ秒の区間を等分割した．TCP スループットは 100MB のファイルを 10 回伝送した時の中央値である．

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あり，３倍の TCP スループットの向上となった．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が示された．

4. 遅延シミュレータ環境における評価

仮想化環境と実環境では多くの違いがある．特に，TCP スループットに大きな影響を与えると考えられる違いは仮想 NIC である．仮想マシン間の通信はメモリ内で処理されるため，TCP スループットは 400 MB/s(3.2 Gbps) 以上のかなり大きな値であった．このため，仮想化環境で示された TCP スループットの向上が実サーバでも可能であるか評価しなければならない．

そこで本研究では，次に，プロキシとして実サーバを用いた評価を行った．実サーバでの評価にあたり，実サーバ間の往復遅延時間の設定のため，コネクション分割に対応した遅延シミュレータ環境 (図 2) を構築した．

4.1 遅延シミュレータ

本研究で用いる遅延シミュレータは，4 ポートの Gigabit NIC を 2 つ，合計 8 ポートの 1000Base-T ポートを有する

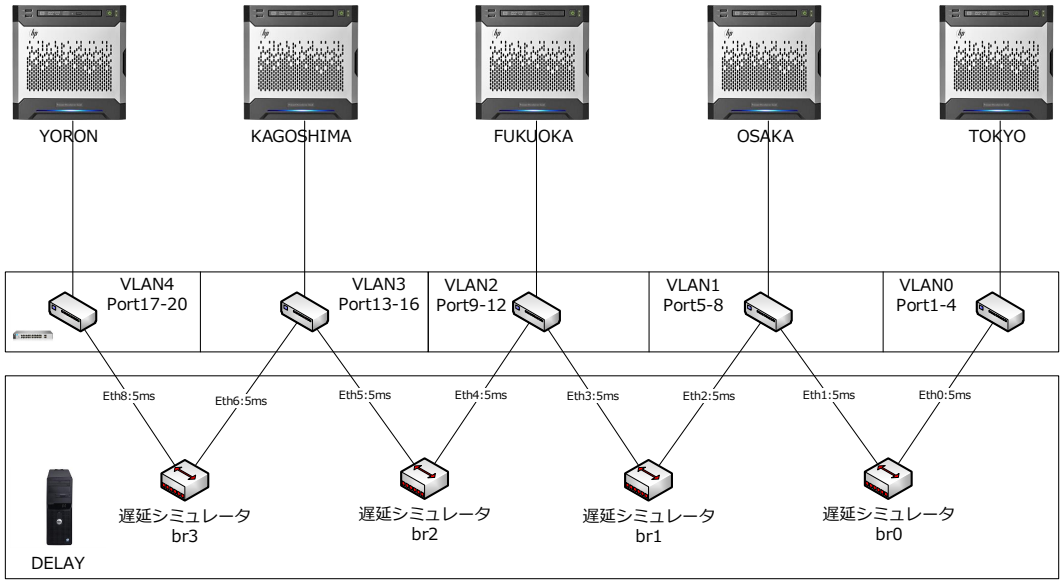

図 2 往復遅延時間が 40 ミリ秒の区間を 4 区間にコネクション分割する場合の遅延シミュレータの構成図．各ポートに 5 ミリ秒の遅延を設定することで，各区間の往復遅延時間を 10 ミリ秒，合計の往復遅延時間を 40 ミリ秒に設定できる．

サーバと，サーバに接続したレイヤ 2 スイッチにより構成した．遅延の設定は遅延シミュレータのサーバ上で netem を用いて行う．測定の対象となるサーバ，クライアント，プロキシでは，遅延の設定を行わず，レイヤ 2 スイッチ上でそれぞれ独立した VLAN を設定したポートに接続した (図 2)．

サーバでは 4 つのブリッジを設け，それぞれのブリッジに 2 つずつネットワークポートを設定し，netem を用いて任意の遅延を設定できるようにした．これにより，4 区間までのコネクション分割シミュレーションを実現できる．3 区間以下のコネクション分割の場合には，シミュレータの一部のみを利用する．また，この遅延シミュレータを用いればクライアント–サーバ間の経路上にプロキシを設置できる場合のインターネット環境も再現できることになる．

4.2 コネクション分割数と TCP スループット

遅延シミュレータ環境において，TCP プロキシソフトウェアとして dante と squid を用いた FTP スループットと HTTP スループットの測定を行った．コネクション分割数は 1 から 4 までとした．プロキシとして用いたサーバは CentOS 6.5 x86_64 を OS として導入した HP ProLiant MicroServer Gen8(Pentium G2020T, メモリ 4GB) である．サーバ–クライアント間の合計の往復遅延時間は 50 ミリ秒とし，それぞれの区間の往復遅延時間が等しくなるように分割した．分割数 1 は分割しないことを示しており，この場合はプロキシを経由しない．測定に際しては 64 MiB のファイルをダウンロードして TCP スループットを求め，10 回の中央値を TCP スループットとした (図 3)．

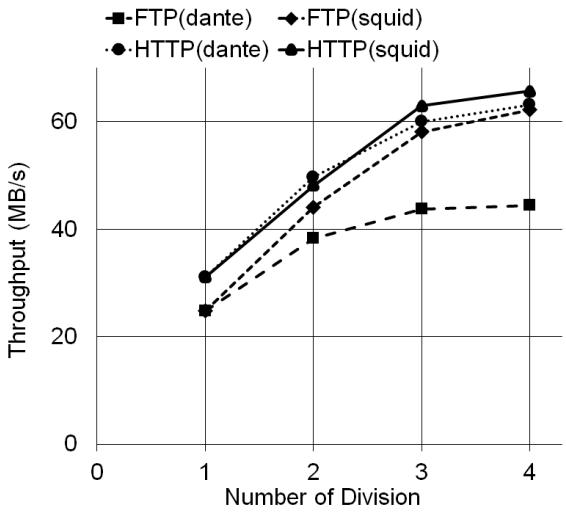

図 3 遅延シミュレータ環境におけるコネクション分割数と TCP スループット (MB/s)．往復遅延時間 50 ミリ秒の区間を等分割しており，TCP スループットは 64 MiB のファイルを 10 回伝送した時の中央値である．

プロキシに dante を用いた場合の FTP プロトコルの TCP スループット向上率が低いが，その他のプロキシとプロトコルの組み合わせでは，TCP スループットの向上は同じ傾向となった．HTTP プロトコルでは dante と squid に大きな差は見られない．

向上率で見ると表 2 のようになる．2 分割時の向上率は最大 178%となっており，表 1 に示す仮想化環境における 2 分割時の値である 170%と同等の TCP スループット向上となった．一方，4 分割では最大 2.5 倍となっており，4 分割で 3 倍の向上が見られた仮想化環境に比べると，TCP

スループット向上の効果が低くなった.

表 2 分割数と TCP スループット向上率の最小・最大．プロトコル-プロキシの組み合わせを () 内に示す.

分割数	最小向上率	最大向上率
2	154%(HTTP-squid)	178%(FTP-squid)
3	176%(FTP-dante)	234%(FTP-squid)
4	179%(FTP-dante)	251%(FTP-squid)

遅延シミュレータ環境において分割数を増やした場合，仮想化環境に比べて向上率が低下する．しかし TCP スループットは向上しており，実サーバを用いた場合でも，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が示された.

5. インターネット環境における評価

仮想化環境，遅延シミュレータ環境とも，コネクション分割により TCP スループットが向上することが示された．そこで本研究では，最後に，インターネット環境において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有効性を検証した.

5.1 サーバ及びプロキシの配置

インターネット環境においてコネクション分割による TCP スループットの向上効果を得られるかを検証するため，東京に測定対象サーバ，大阪，福岡，鹿児島にプロキシをそれぞれ置き，鹿児島県与論町に設置した測定用クライアントからプロキシを経由して TCP スループットの測定を行った.

東京のサーバはさくらインターネットの「さくらの VPS」4G プランを用い，仮想 4 コア，メモリ 4GB の仮想マシンに Oracle Linux 6.5 x86_64 を導入して用いた．Oracle Linux は CentOS と同様，Red Hat Enterprise Linux 互換ディストリビューションであるが独自の Linux カーネル (UEK: Unbreakable Enterprise Kernel) を標準で搭載しており，カーネルバージョンは 3.8.13 である．さくらの VPS ではインターネット接続回線は 100Mbps の共有回線であると公表されている.

大阪と福岡では市内のデータセンターに，鹿児島では鹿児島大学内にサーバ (HP ProLiant ML320e Gen8 v2, Xeon E3-1240v3, 8GB メモリ) を設置し，Oracle Linux 6.5 x86_64 を導入してプロキシとして用いた.

プロキシと測定用クライアントのインターネット接続回線にはフレッツ光ネクスト隼 (アクセス回線の速度は 1Gbps) を用い，ISP として OCN を利用した．インターネット接続用のルータはいずれも NEC UNIVERGE IX2105 である.

それぞれの拠点間の往復遅延時間は表 3 の通りとなった．プロキシを経由しない場合に比べて，プロキシを経由する場合の往復遅延時間が大きくなっている．これは与論から東京までの経路上にプロキシを設置することができていないためと考えられる.

表 3 プロキシ位置と往復遅延時間 (ミリ秒)

プロキシ位置	与論–プロキシ	プロキシ–東京	与論–東京
なし			38.6
鹿児島	31.8	21.2	53.0
福岡	26.6	26.4	53.0
大阪	30.5	12.6	43.1

また，このことから，コネクション分割は大阪，福岡，鹿児島のプロキシによる２分割の３通りを評価することにした．これは，３分割以上にした場合，合計の往復遅延時間が大きくなり，区間分割によるスループット向上効果を期待できないためである.

仮想化環境や遅延シミュレータ環境とは異なり，インターネット環境ではサーバとクライアント間の任意の位置にプロキシを設置することは難しい．異なる ISP を利用する場合，ISP 同士が接続している IX 経由となるため，ほとんどの場合大回りの接続となり，区間分割を実現することができない．また，同一の ISP であっても複数の経路を有している場合には，サーバに直接接続する場合とプロキシを経由した通信の経路が異なる場合がある．本研究では，サーバへの経路の中間の位置にプロキシを置く形になることを期待してプロキシとクライアントについて同じ ISP を利用したが，プロキシ経由の場合に往復遅延時間が大きくなっており，利用した ISP では直接通信の場合と異なる経路になっていると思われる.

5.2 プロキシ位置と TCP スループット

インターネットで最も一般的な利用形態である HTTP プロトコルについて，TCP スループット向上効果を確認するため，TCP プロキシソフトウェアとして squid を用いた HTTP スループットの測定を，プロキシ位置を変えて行った．測定に際しては 8 MiB のファイルを 10 回ダウンロードし，中央値をスループットとした (表 4).

表 4 プロキシ位置と TCP スループット (Mbps)

プロキシ位置	TCP スループット (Mbps)	向上率 (%)
なし	48.2	
鹿児島	65.8	137
福岡	58.6	122
大阪	48.1	100

仮想化環境や遅延シミュレータ環境に比べて伝送するファイルが小さい．これは，サーバとして用いた VPS のインターネット接続回線が 100 Mbps であるためである．サイズの大きなファイルを伝送した場合，帯域を使い切ってしまい TCP スループットを正しく測定することができなかった.

プロキシ経由の通信は直接通信に比べて合計の往復遅延時間が大きいため，コネクション分割によるスループット向上の効果が無ければ，プロキシを経由することでスループットが低下する．しかし，大阪のプロキシを経由した場合にわずかに TCP スループットが低下したものの，鹿児島，福岡のプロキシを経由することで，TCP スループットの向上が見られ，コネクション分割による TCP スループットの向上を確認できた．

プロキシを経由することで合計の往復遅延時間が大きくなっているため，直接通信に対する向上率は遅延シミュレータによる測定と比べて小さい．しかし，往復遅延時間の増大による TCP スループット低下を上回る向上を達成することができた．これにより，与論からのインターネットアクセスにおいて，鹿児島市内に設置したプロキシを経由することで，HTTP スループットが向上することが明らかとなった．

しかし，インターネットの回線状況は日によって，また，時間帯によっても大きく変動する．ISP が違えば，その傾向も異なるはずである．今回用いた OCN は利用者が多くなる時間帯に帯域が不足する傾向が見られたため，測定は利用者が少ないと思われる時間帯に行った．ただ，実用化を考えれば，混雑時にどの程度の性能を発揮できるかについても検証の必要がある．

5.3 大阪をサーバとした場合の向上率

日本国内ではサーバが東京に集中しているため，「インターネットの速度」は東京に設置されているサーバに対する TCP スループットと同じ意味である場合が多い．しかし，大阪でもハウジングや VPS サービスは提供されている．そこで，大阪のプロキシを測定対象サーバとし，福岡と鹿児島のプロキシを経由して，与論からの測定を行った．向上率を表 5 に示す．さくら VPS と異なりアクセス回線の速度が 1Gbps であることから，伝送サイズが大きい場合でも TCP スループットを測定できた．

表 5 大阪をサーバとしたときの伝送サイズごとの TCP スループット向上率 (%)

伝送サイズ	8MiB	16MiB	32MiB	64MiB	128MiB
鹿児島経由	139.9	121.8	147.8	129.2	147.4
福岡経由	129.5	113.6	123.4	100.1	143.8

この結果より，与論からのアクセスの場合，大阪に置かれているサーバに対しても鹿児島に設置したプロキシが有効であると言える．

6. まとめ

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，仮想化環境及び遅延シミュレータ環境において，TCP 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．また，インターネット環境で TCP プロキシソフトウェアとして squid を用いた場合の HTTP スループットを測定し，本システムがインターネット環境においても有効であることを示した．

TCP コネクションの分割にあたっては，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するが，仮想化環境では４分割，遅延シミュレータ環境では３分割までが効率よく TCP スループットが向上する．

インターネット環境では，等分割になるようにプロキシを設置することが困難であること，プロキシを設置できる場所が最短経路上にあるとは限らないこと，さらにデータセンターなどに設置の場合は施設の費用や回線費用が必要となることから，中間の 1 カ所にプロキシを設置する方法が有効であると考えられる．

本研究では，コネクション分割のみによる TCP スループット向上を図った．PEP で用いられている TCP スループット向上の仕組みを組み合わせることで，より一層の TCP スループット向上も可能になるはずである．

謝辞

本研究は，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．また，測定および分析に協力していただいた鹿児島大学大学院理工学研究科博士前期課程学生の室屋孝英氏に感謝の意を表したい．

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] 升屋正人, 室屋孝英, 下園幸一, 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, 大学情報システム環境研究, Vol. 17, pp. 58-66 (2014)
[5] Allman, M. , Floyd, S. and Partridge, C.: Increasing TCP's Initial Window, RFC 3390 (2002)
[6] Border, J., Kojo, M., Griner, J., Montenegro, G. and Shelby, Z.: Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations, RFC 3135 (2001)
[7] 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)
[8] Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D., and Jones, L.: SOCKS Protocol Version 5, RFC 1928 (1996)