ボトルネックRTTの最小化に基づくTCP分割手法の提案とPlanetlab環境による実証評価

樗林勇気[†1] 中山雅哉[†2]

広帯域長距離ネットワークで TCP を用いてデータ転送を行うと，帯域遅延積や輻輳制御の影響を受けるために，高いスループットを達成することが困難である．この問題を解決する一つの手法として，中継ノードを用いて RTT を小さくする TCP 分割手法が提案されている．TCP 分割手法でより高いスループットを達成するためには，送受信ノードに適した中継ノードの選定が必要である．本研究では，ボトルネックとなる RTT を最小化するように中継ノードを選定する TCP 分割手法を提案する．Planetlab を用いて提案手法を実装し，中継ノードを 1 つのみ用いる手法との比較検討を行った．

A proposal of TCP-splitting method based on minimization of bottleneck RTT and its evaluation using Planetlab environment

Yuki KUREBAYASHI [†1] and Masaya NAKAYAMA[†2]

It is difficult to achieve high throughput data transmission between long distance nodes using TCP applications because of effects of BandDelayProduct and congestion control. One approach to solve this problem is TCP-splitting method by using relay nodes owing to reducing RTT. In order to achieve higher throughput on TCP-splitting method, it takes selections of relay nodes for source node and destination node. We propose a TCP-splitting method to select relay nodes based on minimization of bottleneck RTT. We implemented a proposed method using Planetlab, and weighed a method using only a relay node.

1. はじめに

近年，海外に生産拠点を設置したり，自然災害に備えて遠隔地にバックアップをとるなどグローバルな事業展開をする企業は増える傾向にある．このように遠距離拠点間でのデータの送受信が増える一方で，データ量は肥大化し続けている．そのため，円滑に業務を行うためにも高速なファイル転送の需要が高まっている．また，個人でもクラウドコンピューティングのサービスとしてオンラインストレージを使用している人も増加している．しかしながら, 長距離広帯域ネットワークにおいて, 現在多くのアプリケーションで使用されている TCP(Transmission Control Protocol) では, 高いスループットを達成することが難しいと報告されている．この問題を解決する手法として TCP 分割手法がある．これは，送信ノードと受信ノードの間に中継ノードを用いて本来送信ノードと受信ノード間で張られる TCP コネクションを送信ノードと中継ノード間，および (中継ノードと中継ノード間，) 中継ノードと受信ノード間で張り，RTT を小さくするという手法である．しかしながら，この中継ノードは，送受信ノードに適したノードを選定する必要がある．

本論文では，ボトルネック RTT の最小化に基づき中継ノードを選定する提案方式を Planetlab[1)] を用いて実装した．そして，その実装下で，中継ノードを 1 つのみ選定する手法[13)] との比較検討を行った．その結果，送受信ノードが 200 msec より大きい遅延があるデータ転送においては，提案手法がより高いスループットを達成可能であると予測することができた．

本論文の構成は以下の通りである．2 章では，TCP 分割手法について触れる．3 章では，関連研究について触れる．また，送受信ノードに適した中継ノードを選定するという本研究の目的を述べる．4 章では，ボトルネック RTT の最小化に基づく中継ノードの選定という本研究の提案手法について述べる．5 章と 6 章では，提案手法の実装と，その実装下で中継ノードを 1 つのみ用いた場合との比較評価について述べる．最後に 7 章で本論文の結論と今後の課題について述べる．

†1 東京大学大学院工学系研究科電気系工学専攻
Department of Electrical Engineering and Information Systems,Graduate School of Engineering,The University of Tokyo
†2 東京大学情報基盤センター
Information Technology Center, The University of Tokyo

2. TCP 分割手法

2.1 TCP の問題

TCP を用いたアプリケーションで高いスループットを達成することが難しい理由は，TCP が帯域遅延積 (BDP:Bandwidth Delay Product) や輻輳制御の影響を受けるためである．帯域遅延積とは，TCP を利用してデータ転送を行なう際に，送信ノードと受信ノードのリンク間帯域と，送信ノードと受信ノード間の RTT[⋆1]，受信側の広告ウィンドウ長 (RWIN:Receive Window) もしくは MSS[⋆2]によって，送信ノードと受信ノードのリンク間で実現可能な最大スループットが決定される法則であり，スループットは，式 (1) で表わされる．p は パケットロス率である．つまり，スループットは RTT の影響を受け，RTT が増加するほど，スループットは低下することになる．なお，正確なスループットの計算に関しては，[9] を参照されたい．

$$Throughput = \frac{6\sqrt{2}MSS}{4\sqrt{3p}RTT} \qquad (1)$$

2.2 TCP 分割手法

2.1 節での問題を解決するために，MTU [⋆3]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，スループットの向上が期待できる．一方で，経路の途中で TCP コネクションを分割する手法が TCP 分割手法[2] である．なお，TCP プロキシと呼ばれることもあるが，本稿では TCP 分割手法として表記する．**図 1** のように TCP コネクションを送受信ノード間の経路上にある中継ノードで終端することによって TCP コネクションを複数に分割する手法である．分割された TCP コネクション毎にパケットを中継しながらデータ転送を行う．送信ノードからパケットが送信されて中継ノードが受信すると，中継ノードはそのパケットを受信ノードに転送する．また，TCP 分割手法では，中継ノードはパケットを転送するだけではなく，受信ノードに代わって ACK を返す機能を有している．そのため，送信ノードは，受信ノードからの ACK を待つことなく，中継ノードからの ACK により新たなデータパケットの送信を行うことが出来る．これにより，送信ノードと中継ノードの間の RTT を小さくすることが出来る．同様に，中継ノードと受信ノードとの間の RTT も小さくすることが出来る．結果として送信ノードと受信ノード間の RTT を小さくすることが出来る．また，中継ノードは，データパケットに対する ACK を受信するまで，データパケットをバッファリングしているので再送要求にも応えることが出来る．

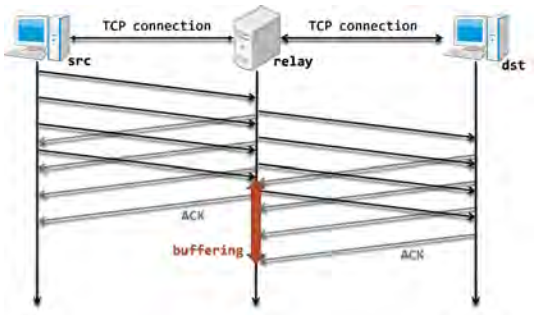

図 1 TCP 分割手法

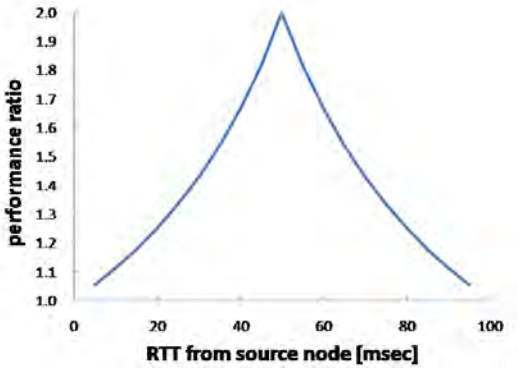

図 2 TCP 分割手法によるスループットの向上

式 (1) でスループットが計算できるため，RTT を等分割できるような中継ノードが存在していれば，スループットは 2 倍の向上が見込める．しかしながら，RTT を等分割できない場合は，一番大きい RTT が送信ノードと受信ノード間の RTT で支配的になる．送信ノードから受信ノードまでの RTT が 100 msec のときに，中継ノードの位置を変化させたときの式 (1) に基づいたスループットの理論値を**図 2** に示す．横軸は送信ノードから中継ノードまでの RTT で，縦軸は中継のオーバーヘッドがない場合のスループットの性能向上率であり，式 (2) で求める．ここで，RTT_{direct} は送信ノードと受信ノード間の RTT で，$RTT_{bottleneck}$ は分割した TCP コネクションのうち最も大きい RTT である．

$$\text{performance ratio} = \frac{RTT_{direct}}{RTT_{bottleneck}} \qquad (2)$$

3. 関 連 研 究

TCP 分割手法の関連研究として，衛星回線に適応した TCP 分割機構[6]，ネットワーク状況に応じたス

⋆1 Round Trip Time：エンドノード間でパケットが往復するのに要する時間．TCP では，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・変動は経路の混雑に影響される．

⋆2 Maximum Segment Size：受信可能なセグメントサイズの最大値である．

⋆3 Maximum Transfer Unit：あるリンク上で転送可能な最大の IP パケット長．

ループット解析[12)8)11)5)]，中継ノードの両端のネットワーク状況に応じて TCP のコネクション数を変える手法[4)]，中継ノードを 1 つ用いて RTT を分割する手法[13)] などがある．

Luglio らは，衛星回線を対象とした TCP 分割手法を提案している[6)]．これは，衛星との角度等を算出したモデルで実際のネットワークに適応できるものではない．

ネットワーク状況に応じたスループット解析では，磯垣らは，送信バッファが通信に与える影響の解析[12)]，Maki らは，中継ノードの送信バッファが空になったとき通信が一時的に停止したときの状況を想定したスループット解析[8)] を行っている．これらは中継ノードの両端のネットワーク状況が異なると，受信バッファと送信バッファに差が生じる問題の影響を調査するためである．これらの文献は，送信バッファを大きくすることでこの低下を抑えられると述べている．また，Wolf，Ladiwala らは，中継ノードを複数用いた場合，ウィンドウサイズを変化させた場合，パケットロス率を変化させた場合のネットワーク状況でのシミュレーションを行っている[11)5)]．しかしながら，中継ノードは等間隔に分割できることを前提としているために実ネットワークでの実現は難しいと考えられる．

Karbhari らは，両端のネットワーク状況の差によって送信バッファと受信バッファに差が生じる問題を解決するために，ホップ数に応じて TCP コネクションをパラレルに張り，円滑に中継する手法を提案している[4)]．これにより，スループットが向上することを示している．しかしながら，中継ノードの選定法については述べておらず，中継ノードの両端のネットワークの差による問題の解決法のみを示している．

小林らは，中継ノードを 1 つ用いた中継ノードの選定法を提案している[13)]．これは，送信ノードと中継ノード間の RTT と中継ノードと受信ノード間の RTT のうち大きい方の RTT が，送信ノードと受信ノード間の RTT よりも小さくなる中継ノードを中継候補ノードから 1 つ選定するというものである．Ladiwala らの研究では等間隔の分割を前提とはしていたものの，中継ノードの数が多い方がボトルネック RTT を小さくできる可能性が高い．そのため，中継候補ノードから 1 つのみ選定する手法では，ボトルネック RTT を最小化は出来ているとは言えない．そこで，複数の中継ノードを用いてボトルネック RTT を最小化することで送信ノードと受信ノードに適した中継ノードを選定できると考えられる．

4. 提 案 手 法

送信ノードと受信ノードに適した中継ノードを選定するのが本研究の目的である．本論文では，式 (1) の RTT に着目し，ボトルネック RTT を最小化することが出来る中継ノードを選定する．

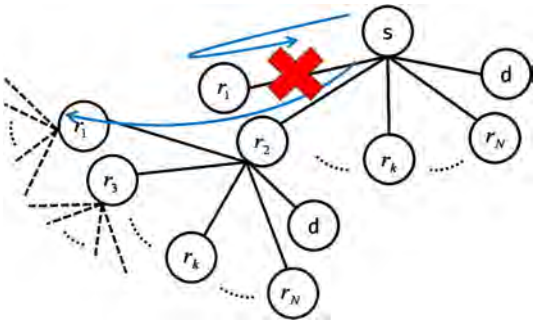

図 4 解析手順 (1)-(2)

4.1 用語の定義

まずは，用語の定義を行う．

- RTT_{xy}：ノード x からノード y までの traceroute の結果による RTT.
- s：送信ノード.
- d：受信ノード.
- $r_i (1 \leq i \leq N)$：中継候補ノード.

4.2 中継ノード選定までの手順

次の手順により，中継候補ノードから中継ノードを選定する．

(1) 送信ノード s は，多数の計算機に対して命令を実行できるグリッド用シェル GXP[10)] を用いて，全ての中継候補ノード $\{r_i\}$ に，traceroute の実行要求を送る．

(2) 送信ノード s，および実行要求を受けた中継候補ノード $\{r_i\}$ は，受信ノード d および全ての中継候補ノード $\{r_i\}$ に対して traceroute を実行し，その結果を保持する．

(3) 全ての中継候補ノード $\{r_i\}$ は，traceroute が終わり次第，保持した結果を送信ノード s に転送する．

(4) 送信ノード s は，自ノードの結果および全ての中継候補ノード $\{r_i\}$ からの結果を用いて解析を行い，中継ノードを選定する．

4.3 解　　析

解析手法について説明する．基本は深さ優先探索を用いて探索する．目的はボトルネック RTT の最小化であるので，1 つでも $RTT_{jk} > RTT_{sd}$ となるようなホップが存在する場合は適していない．そこで，閾値 th を用意する．なお，th の初期値は，RTT_{sd} である．

(1) 送信ノード s は，全ての中継ノード $\{r_i\}$ および受信ノード d 含むデータ軍 $\{n_j\}$ に対して順に，RTT_{sn_k} と th をと比較する．(**図 4**)

(2) $RTT_{sn_k} < th$ の場合，その n_k を保持し，(1) を送信ノード s の代わりに n_k として，繰り返す．なお，保持しているノードを $\{n_j\}$ から省きながら，順次探索する．(図 4)

(3) 順次に探索していき受信ノード d が選ばれた場合，その経路での最大の RTT を閾値 th とし，その時のノードの組み合わせを保持する (**図 5**).

(4) 閾値 th の更新を繰り返しながら，深さ優先探索

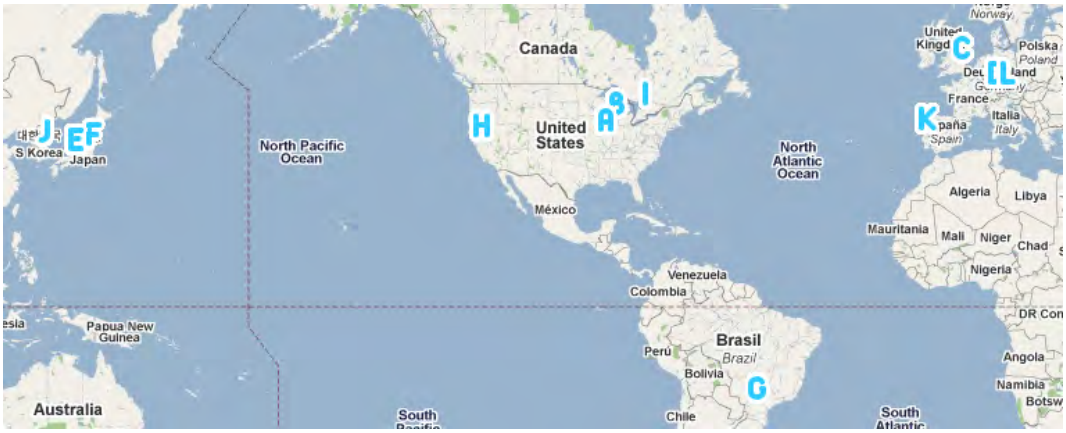

図 3 ノード位置

表 1 ノード情報

	node	IP address	location
A	vn1.cs.wustl.edu	128.252.19.20	St.Louis (U.S.)
B	planetlab1.cs.uchicago.edu	128.135.11.149	Chicago (U.S.)
C	planetlab1.cs.ucl.ac.uk	193.63.58.70	London (U.K.)
D	planetlab2.csg.uzh.ch	192.41.135.219	Zurich (Switzerland)
E	planetlab0.dojima.wide.ad.jp	203.178.133.10	Dojima (Japan)
F	planetlab0.otemachi.wide.ad.jp	203.178.133.2	Otemachi (Japan)
G	planetlab1.c3sl.ufpr.br	200.17.202.194	Parana (Brazil)
H	planetlabnode-1.docomolabs-usa.com	216.98.102.29	Los Angels (U.S.)
I	plonk.cs.uwaterloo.ca	129.97.74.14	Waterloo (Canada)
J	csplanetlab3.kaist.ac.kr	143.248.139.56	Daejeon (Korea)
K	planetlab-1.iscte.pt	193.136.191.25	Lisboa (Portugal)
L	plab1-c703.uibk.ac.at	138.232.66.194	Innsbruck (Austria)

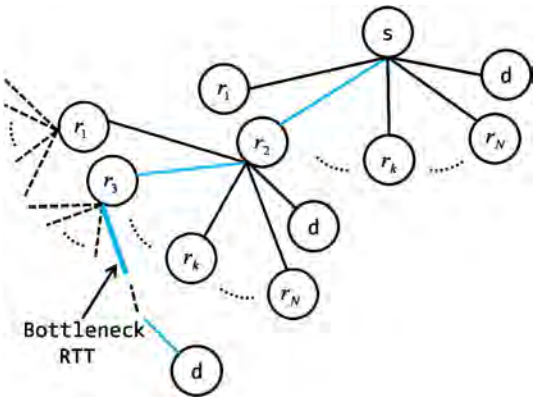

図 5 解析手順 (3)

を繰り返していき，全てが終わった時点での組み合わせの中継候補ノードを中継ノードとする．

5. 実　　装

本実装では，Planetlab[1] を利用した．Planetlab とは，オーバーレイネットワークに関するテストベッド環境である．世界中の研究，教育機関が，それぞれが持つ計算資源を提供することによって成り立っている環境である．12 ノードをノード群として利用した．利用したノードの情報を**表 1** および**図 3** に示す．本実装は，ノード群から送信ノードと受信ノードを 1 つずつ決定した時に，残りのノード群を中継候補ノードとした ${}_{12}P_2 = 132$ 通りである．送信ノード s はこれら中継候補ノード $\{r_i\}$ のうちから中継ノードを選定し，TCP 分割を行う．今後も，PlanetLab 上のノードを中継ノードとして用いて TCP 分割を行うことを想定し，TCP 分割のノード群の情報はレポジトリが管理していることを予定している．

ノード群に新しいノードが追加された場合，レポジトリはノード群に対し，ブロードキャストを行いノード情報の更新を行う．本稿では，ノード群は最新のノード情報を持っているものと仮定して実験を行っている．**図 6** を含むプログラムを全ノードに実装した．RTT の値は，UTC 09/09/29 9:00 における traceroute の結果を用いた．

6. 評　　価

6.1 RTT_{direct} との比較

提案手法により $RTT_{bottleneck}$ がどの程度小さくできたかを示す．

RTT_{direct} と提案手法による $RTT_{bottleneck}$ の関係

```
for($j[1]=$N+1; $j[1]>0; $j[1]--){
  &DFS($s, $n[$j[1]], 1);
}
sub DFS
  my($x, $y, $z) = @_;
  for($k=1; $k<=$z; $k++){
    if($relay[$k] eq $y){
      return;
    }
  }
  if($RTT_th > &RTT_xy($x,$y)){
    if($y ne $d){
      $relay[$z] = $y;
      for($j[$z+1] = $N+1; $j[$z+1] > 0; $j[$z+1]--){
        &DFS($y, $n[$j[$z+1]], $z+1);
      }
    $relay[$z] = ();
    }else{
      $relay[$z] = $y;
      $RTT_bottleneck = &max_RTT(@relay);
      if($RTT_bottleneck < $RTT_th){
        $RTT_th = $RTT_bottleneck;
        @opt = @relay;
        $relay[$z] = ();
      }
    }
  }
}
```

図 6 枝刈り深さ優先探索ソースプログラム

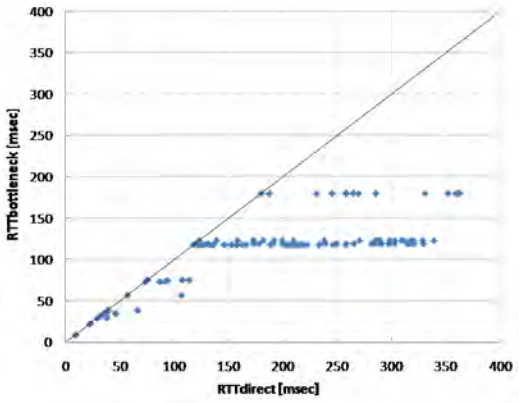

図 7 RTT_{direct} と提案手法の $RTT_{bottleneck}$ の関係

を**図 7**，RTT_{direct} と提案手法の性能向上率の関係を**図 8** に示す．図 7，図 8 より RTT_{direct} が 75 msec までは，RTT_{direct}=$RTT_{bottleneck}$ の直線に沿うものが多く，performance ratio もほとんどは 1 である．これは送信ノード s と受信ノード d の RTT を分割するような場所に中継ノードが殆ど存在せず，中継ノードが選定されていないからである．RTT_{direct} が 75 msec 以上になると，RTT_{direct}=$RTT_{bottleneck}$ の直線に沿うものは少なく，多くの場合で，performance ratio もほとんどは 1 を超えた．これは送信ノード s と受信ノード d の距離が長くなり，RTT を分割することができる中継候補ノードが増えたためである．また，RTT_{direct} が 75 msec 以上のプロットは，$RTT_{bottleneck}$ は大きく分けて 125 msec の層と 175 msec の層に分けることができる．これは海を跨ぐホップで，$RTT_{bottleneck}$ が 125msec のときは北太平洋をホップする場合であり，175msec のときはアメリカ大陸と南アメリカ大陸のホップの場合である．海の上に中継候補ノードが存

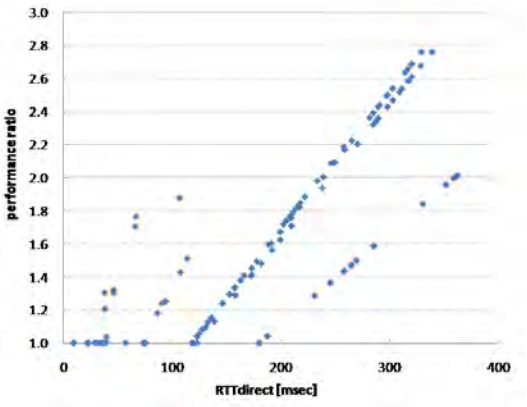

図 8 RTT_{direct} と提案手法の性能向上率の関係

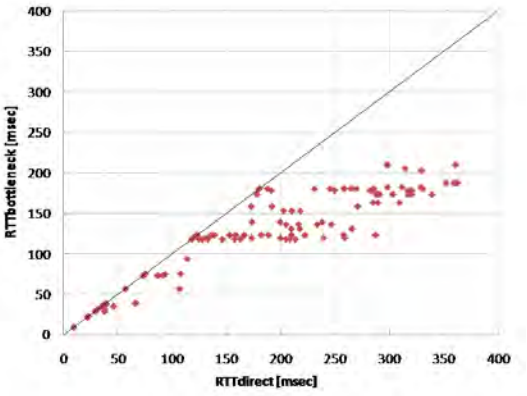

図 9 RTT_{direct} と既存手法の $RTT_{bottleneck}$ の関係

在しないために，海を跨ぐホップの RTT を小さくすることは難しい．しかしながら，中継ノードが増えると，例えば，日本からヨーロッパでの通信のときには、太平洋をホップせずに陸続きで中継する等の海を跨がないような中継の組み合わせが期待できる．

6.2 既存手法との比較

比較対象として，小林ら[13] の既存手法と比較する．全ての中継候補ノード $\{r_i\}$ に対して，それぞれ $max(RTT_{sr_k}, RTT_{r_kd})$ を算出し，RTT_{sd} を含めた全通りを比較する．その中で最小となるときの r_k を中継ノードとして選定する．RTT_{sd} となる場合は中継ノードはない．

RTT_{direct} と既存手法による $RTT_{bottleneck}$ の関係を**図 9**，RTT_{direct} と既存手法の性能向上率の関係を**図 10** に示す．図 7 と図 9，図 8 と図 10 をそれぞれ比較すると，RTT_{dicret} が 150 msec 以下ではほとんど変化が見られない．これは中継ノードがない，もしくは 1 つであるために変化が生じない．しかしながら，RTT_{dicret} が 200 msec を超えると，既存手法では，前述の $RTT_{bottleneck}$ が 125 msec 付近のときの層が疎らになっているのがわかる．これは提案手法では中継ノードでは中継ノードを 2 つ以上用いて，一番小さな太平洋のホップに抑えているのに対して，既存

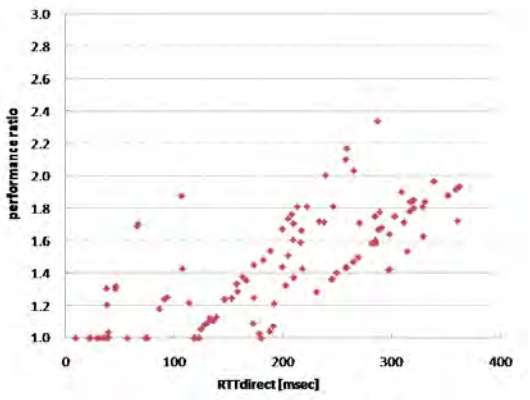

図 10 RTT_{direct} と既存手法の性能向上率の関係

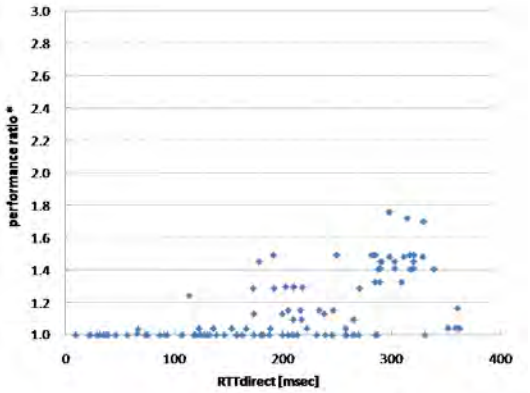

図 11 RTT_{direct} と既存手法に対しての提案手法の性能向上率の関係

手法では 1 つしか中継ノードを選定していないために大きくなっている．

$RTT_{bottleneck}$ と既存手法に対しての提案手法の性能向上率の関係を**図 11** に示す．なお，$performanceratio^*$ は式 (3) で表わされ，$RTT^{existing}_{bottleneck}$ は既存手法の $RTT_{bottleneck}$，$RTT^{proposed}_{bottleneck}$ は提案手法の $RTT_{bottleneck}$ である．

$$performanceratio^* = \frac{RTT^{existing}_{bottleneck}}{RTT^{proposed}_{bottleneck}} \qquad (3)$$

図 11 より，RTT_{direct} が 200 msec 以上になると中継ノードを 2 つ以上用いる有効性が出始める．特に RTT_{direct} が 300 msec 付近では，殆どの場合において 2 つ以上用いてボトルネック RTT を最小化しており，既存手法に対してもさらに 1.5 倍程度の性能向上率が期待できる．RTT_{direct} が 400 msec 付近の場合は，G:planetlab1.c3sl.ufpr.br，とアジア圏のノードとの通信で，太平洋提案手法，既存手法どちらも RTT_{BG} が $RTT_{bottleneck}$ となっており，南アメリカ大陸にノードが少ないために，そのような結果になったと考えられる．

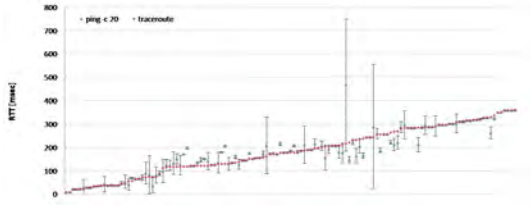

図 12 ping と traceroute の比較

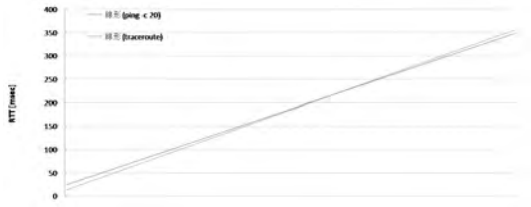

図 13 ping と traceroute の線形近似比較

7. 結論と今後の課題

本研究において我々は，ボトルネック RTT の最小化に基づく中継ノードの選定手法を提案した．提案手法により，点在している中継候補ノードから効率的に中継ノードを選定し，TCP 分割手法を行うことができる．

また，本論文では提案手法を Planetlab を用いて実装を行うとともに，既存手法である 1 つの中継ノードを用いる手法と比較検討を行った．そこで，実際に TCP 分割手法を用いるときに必要になる中継候補ノードを実ネットワーク上に用意した．これは，中継ノードがあるものだと仮定してシミュレーションを行っている関連研究と比較した場合に実用的であると考えられる．実証により，100 msec 以上では TCP 分割手法の有効性が，200 msec 以上では，中継ノードを 2 つ以上用いてボトルネック RTT を最小化する提案手法の有効性が示すことができた．

今後の課題としては，現在の手法では，ボトルネック RTT を最小化することのみを優先して考慮しているために，余分な中継ノードが存在する場合がある．これは，[4)5)8)] でも述べられているようにバッファの問題にも関わるため，余分な中継ノードを削減することが課題である．また，中継する際のオーバヘッドの検討も行う予定である．本論文では，ボトルネック RTT の最小化に基づいて中継ノードの選定を行った．さらに，ネットワーク帯域等の他のネットワーク状況を考慮する手法や Vivaldi[3)], Lighthouse[7)] 等のネットワーク座標系の研究との比較検討を行う必要もある．リポジトリがノード群に対してノード情報の更新を行うシステムの実装も今後の課題と言える．

付　　録

A.1　ping と traceroute の比較

本稿では，RTT の測定に traceroute を用いている．ping を用いて測定した RTT と traceroute を用いて測定した RTT を比較する．両方とも，UTC 09/09/29 9:00 におけるデータを用いた．traceroute の測定が終了後，ping の測定を開始した．

traceroute を用いた場合と，ping -c 20 を用いた場合のそれぞれの送受信ノードの組の RTT の結果を**図 12** に示す．ping の場合は，95%信頼区間を併せて示す．また，**図 13** に線形近似の比較を示す．図 13 より，ping と traceroute で，RTT に大きな差異はそこまで見られない．

参　考　文　献

1) : Planetlab, *https://www.planet-lab.org/*.
2) Cohen., R. and Ramanathan., S.: Using proxies to enhance TCP performance over hybrid fiber coaxial networks, *Computer Communications*, Vol.20, pp.1502–1518(17) (1998).
3) Cox, R., Dabek, F., Kaashoek, F., Li, J. and Morris, R.: Practical, distributed network coordinates, *SIGCOMM Comput. Commun. Rev.*, Vol.34, No.1, pp.113–118 (2004).
4) Karbhari, P., Ammar, M. and Zegura, E.: Optimizing End-to-End Throughput for Data Transfers on an Overlay-TCP Path, *Proceedings of Networking*, Springer (2005).
5) Ladiwala, S., Ramaswamy, R. and Wolf, T.: Transparent TCP acceleration, *Comput. Commun.*, Vol.32, No.4, pp.691–702 (2009).
6) Luglio, M., Sanadidi, M., Gerla, M. and Stepanek, J.: On-board satellite "split TCP" proxy, *IEEE Journal on Selected Areas in Communications*, Vol. 22, No. 2, pp. 362–370 (2004).
7) M.Pias, J.Crowcroft, S. W. T.H. and Bhatti, S.: Lighthouses for scalable distributed location, *IPTPS 2003*, pp.278–326 (2003).
8) Maki, I., Hasegawa, G., Murata, M. and Murase, T.: Performance analysis and improvement of TCP proxy mechanism in TCP overlay networks, *Communications, 2005. ICC 2005. 2005 IEEE International Conference on*, Vol.1 (2005).
9) Padhye, J., Firoiu, V., Towsley, D. and Kurose, J.: Modeling TCP throughput: a simple model and its empirical validation, *SIGCOMM Comput. Commun. Rev.*, Vol.28, No.4, pp.303–314 (1998).
10) Taura, K.: GXP : An Interactive Shell for the Grid Environment, *Information Assurance, IEEE International Workshop on/Innovative Architecture for Future Generation High-Performance Processors and Systems, International Workshop on/Innovative Architecture, International Workshop on*, Vol.0, pp.59–67 (2004).
11) Wolf, T., You, S. and Ramaswamy, R.: Transparent TCP Acceleration Through Network Processing, pp.750–754 (2005).
12) 磯垣　順，井口　寧：Improvement of Reno's Performance using Conversion Technique into HighSpeed TCP (2005).
13) 小林弘和，中山雅哉：中継ノードを利用した長距離 TCP データ転送の効率向上と中継ノード発見手法の比較 (インターネット技術及び一般 I)，電子情報通信学会技術研究報告. IA, インターネットアーキテクチャ， Vol.107, No.74, pp.17–22 (20070522).