Vol.2015-OS-134 No.16
2015/8/5

Linux MPTCP kernel の ネットワークスループット予測に関する研究

浜崎 拓也[1,a)]　阿部 洋丈[,2,b)]　加藤 和彦[,2,c)]

概要：近年，複数の経路で TCP コネクションを確立可能な Multipath TCP （MPTCP） が注目されている．マルチホーム環境や Software-Defined Network （SDN） 環境では，MPTCP を用いることで複数の経路を活用できるため，高速なデータ転送が期待できる．データ転送における各経路の性能に応じたスケジューリングや経路選択には，ネットワークスループット予測が有効である．柔軟な経路制御が可能な SDN 環境では特に重要であると考えられるが，MPTCP に着目した予測手法はこれまで提案されていない．本稿では，TCP を想定した既存予測手法の適用可能性の検証結果を報告し，MPTCP の特性を考慮した予測手法を提案する．Linux kernel での MPTCP 実装を用いた検証によって，既存手法で用いられる単一の *probe* では，MPTCP コネクションの確立方法が原因となり各経路の性能を正しく計測できないことがわかった．そのため，既存手法を適用した場合は予測精度が低下する．提案手法では，独立した複数の probe を送信することでこの問題に対処し，Support Vector Regression による機械学習を用いて予測を行う．実験用ネットワークでの評価実験によって，提案手法を用いることで既存手法よりも予測精度が向上し，RMSE が最大 29.1 % 減少することを示した．

キーワード：Multipath TCP （MPTCP）, マルチパス転送，ネットワークスループット予測，Support Vector Regression （SVR），Software-Defined Network（SDN）

A study on prediction of network throughput by Linux MPTCP kernel

Takuya Hamazaki[1,a)]　Hirotake Abe[,2,b)]　Kazuhiko Kato[,2,c)]

Abstract: Multipath TCP (MPTCP), a transport protocol that enables network hosts to establish connections over multiple paths, has been gathering attention in recent years. We can transfer data at higher throughput by MPTCP on multi-homing or Software-Defined Networks (SDN). In data transfer, especially on SDN where we can control routing policies flexibly, predicting network throughput is significant for scheduling or selection of paths depending on the performances of those. However, no methods for MPTCP have been proposed yet. In this paper, we report the result of an attempt applying a prediction method for TCP to MPTCP, and propose a extended prediction method for MPTCP taking an MPTCP characteristic into account. Through the experiments by Linux MPTCP kernel, which is an implementation of MPTCP in Linux kernel, we found that a known prediction method for TCP that uses a single *probe* transfer tends to fail to measure path performances in MPTCP transfer and drops prediction accuracy, and that that is caused by a mechanism of how MPTCP establishes multiple connections. Our method uses multiple probes to deal with the problem and predicts by machine learning with Support Vector Regression. We show that our method achieves higher prediction accuracy and decreases the RMSE by 29.1 % at the most.

Keywords: Multipath TCP (MPTCP), Multipath Transfer, Network Throughput Prediction, Support Vector Regression (SVR), Software-Defined Network (SDN)

1. はじめに

自然災害や火災などから重要なデータや IT システムを保護して迅速な復旧を行うため，ディザスタリカバリの必要性が広く認識されている．ディザスタリカバリの具体的手法としてレプリケーションが挙げられる．レプリケーションでは，広域ネットワーク環境を活用して遠隔拠点のサーバにリアルタイムでデータを複製し，障害発生時には遠隔拠点のサーバでシステムを継続する．これにより，地震などによるデータセンタ規模での大規模な障害発生時にもシステムの継続が可能になる．しかし，レプリケーションでは広域のネットワーク環境を利用するため，ネットワーク性能がボトルネックとなり十分に複製を作成できない可能性がある．つまり，どの時点までのデータを復旧できるかを示す指標である RPO （Recovery Objective Point）が増大し，障害時に損失するデータの範囲が大きくなる．そのため，広域ネットワーク環境でのレプリケーションではネットワークスループットの向上が課題となっている [10].

スループットを向上させる方法として，MPTCP (Multipath TCP) [15] を用いたマルチパス TCP 転送がある．MPTCP は TCP の拡張プロトコルであり，複数の経路で TCP コネクションを確立できる，そのため，マルチホーム環境ではネットワークリソースを効率的に活用でき，スループットや安定性が向上する．また，MPTCP は既存の TCP 環境で動作するように設計されているため，TCP を想定した既存のネットワークやアプリケーションに変更を加えることなくマルチパス転送を実現できる．

さらに，SDN（Software-Defined Network）技術を用いることで，マルチパス TCP 転送をより活用できると考えられる．SDN はソフトウェアによって動的な制御が可能なネットワークのコンセプトであり，従来の自律分散的なネットワークでは困難であった動的かつ柔軟なネットワーク制御を実現するものとして注目を集めている．SDN を実現する標準の一つである OpenFlow [6] では，ネットワークの機能をデータプレーンとコントロールプレーンに分離し，それぞれの機能を担う OpenFlow スイッチ・OpenFlow コントローラを結ぶ OpenFlow プロトコルを規定している．OpenFlow によって，パケットを転送するスイッチ群の制御方法を外部コントローラから動的に変更できるため，ネットワークの性能に応じて経路を制御することでスループット向上が期待できる．

OpenFlow 環境でのマルチパス TCP 転送では，自由な経路制御が可能な反面，膨大な経路候補の中から実際に使用する経路を適切に選択しなければならず，経路選択手法が必要であることが指摘されている．前田らはこの問題に対し，スループット予測を用いて適切な経路を選択する手法を提案している [11]．しかし，これまで MPTCP に着目したスループット予測手法は提案されていない．MPTCP の拡張元である TCP を想定した手法はこれまで多く存在するが，それらが マルチパス TCP 転送の予測に適用可能かどうか，適用した場合に TCP 転送の予測時と同等の精度で予測可能かどうかについてはわかっていない．

本研究では，TCP を想定した既存手法の適用可能性の検証と，MPTCP の特性を考慮した予測手法の提案を行った．Linux での MPTCP 実装である Linux MPTCP kernel を用いた検証によって，既存手法で経路性能の計測のために用いられる単一の probe では，MPTCP コネクションの確立方法が原因となり各経路の性能を正しく計測できないことがわかった．そのため，既存手法を適用した場合は予測精度が低下する．提案手法では，独立した複数の probe を用いることでこの問題に対処した．既存手法との比較により，提案手法を用いた場合は予測精度が向上することを確認した．

以降，本稿は以下のように構成されている．第二章では，TCP 転送のスループット予測手法について述べる．第三章では，MPTCP の実装とコネクションの確立方法について述べる．第四章では，本研究の実験環境について述べる．第五章では，既存手法の適用結果を示す．第六章では，提案手法とその評価結果を示す．第七章では，本稿をまとめ，今後の課題について述べる．

2. スループット予測手法

TCP はインターネットをはじめとしたコンピュータネットワークで広く使用されているプロトコルであり，これまで多くのスループット予測手法が提案されている．予測手法には，大きく分けて Formula-based の手法と History-based の手法の二種類がある．Formula-based の手法は，スロースタートや輻輳回避などの TCP の挙動に基づいてスループットを数式化し，計測した帯域幅やパケットロス率，RTT（Round Trip Time: 往復遅延時間）といったいくつかのパスプロパティからスループット予測値を求めるものである [18], [20] ．一方 History-based の手法は，過去のスループットやパスプロパティを記録し，その記録に基づいて経験的に予測を行う．History-based の手法は Forumula-based の手法よりも予測精度が高い傾向にあり，過去の記録を利用できる場合には History-based の手法を用いたほうが良いことが示されている [17].

History-based の手法の一つには，Network Weather Ser-

[1] 筑波大学 システム情報工学研究科
Graduate School of Systems and Information Engineering, University of Tsukuba
[2] 筑波大学 システム情報系
Faculty of Engineering, Information and Systems, University of Tsukuba
[a] hamazaki@osss.cs.tsukuba.ac.jp
[b] habe@cs.tsukuba.ac.jp
[c] kato@cs.tsukuba.ac.jp

vice [22] で使用されている Swany らの手法 [21] がある．この手法では，予測対象の *data* 送信時のスループットと，比較的小さな *probe* 送信時のスループットのそれぞれの CDF（Cumulative Distribution Function: 累積分布関数）を求め，それらの相関を利用して予測を行う．また Lee らによって，CDF の代わりに SVR（Support Vector Regression）[12] による機械学習を用いた手法が提案されている [13]．Lee らの予測手法を以下に示す．

- probe および data を送信し，それぞれのスループット x_i，y_i を計測する．これを繰り返し，n 個の訓練データ (x_i, y_i) $(i = 1, 2, ..., n)$ の組を収集する．
- 収集した訓練データから，SVR を用いた機械学習によって，入力 x_i から y_i を求める予測モデル $f : \mathcal{X} \to \mathcal{Y}$ を構築する．
- 予測時には probe のみを送信し x を得る．x を f に入力し，出力 y を予測値として得る．

SVR は高い汎化能力を持つことが知られており，CDF を用いた手法よりも予測精度が向上することが示されている．また，予測精度の向上がグリッドコンピューティングの性能向上に貢献することも Lee らによるシミュレーションで確認されている [14]．

レプリケーションでは送受信拠点が固定的であるため，過去の記録を利用する History-based の手法が適用可能である．本稿では，Lee らの手法を既存手法として用いた実験結果を示す．

3. Linux MPTCP kernel

3.1 実装

MPTCP は IETF（Internet Engineering Task Force）によって改良が進められている TCP の拡張プロトコルである，[9]． 既存の TCP 環境でそのまま動作するように設計されており，TCP を想定した既存のネットワークやアプリケーションに変更を加えることなく，複数の経路でコネクション（subflow）を確立できる．

Linux MPTCP kernel は Linux カーネルにおける MPTCP 実装であり，ルーヴァンカトリック大学の研究グループによって開発が行われている [16]．現在の実装では TCP の上位レイヤとして実装されており，TCP と共通の socket API を使用する （**図 1**）．

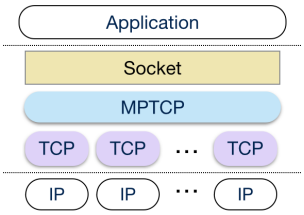

図 1 Linux MPTCP kernel の実装

3.2 MPTCP コネクションの確立

MPTCP は TCP アプリケーションがそのまま動作するように設計されているため，コネクションの開始は TCP と同様に一組の IP アドレス間で行われる．例として，二つのホスト（Client，Server）間での MPTCP コネクションの確立方法を **図 2** に示す．c0, c1, s0, s1 はそれぞれのホストが使用可能な IP アドレスを表しており，c0 - s0 間，c1 - s1 間はそれぞれ異なるネットワークを介して接続されている．また，最初に確立される subflow は，事前にアドレス情報が共有されている c0 - s0 間の subflow 1 とする．このとき，以下に示す方法で MPTCP コネクションが確立される．

- TCP と同様に，3-way handshake によって subflow 1 の確立を開始する (①)．
- このとき Client と Server は，handshake パケットの MP_CAPABLE オプションによって，それぞれ追加の IP アドレス c1，s1 を利用したマルチパス転送が可能なことを相手に通知する．また，追加 subflow 確立時の認証に使用されるランダム生成された key を交換する．
- subflow 1 の確立が完了しデータ転送が開始される (②)．同時に，新たなコネクションを確立する同意が取れる．
- 3-way handshake によって subflow 2 の確立を開始する．このとき，subflow 1 の MPTCP コネクションに参加していることを識別するため，SYN パケット MP_JOIN オプションに key から生成した token を含める．また，SYN/ACK，ACK パケットの MP_JOIN オプションにより，相手が生成したランダムな値 rand と subflow 1 で交換した key から計算した HMAC（Hash-based Message Authentication Code）を交換し，互いに認証を行う．(③)．
- subflow 2 の確立が完了し，subflow 2 でもデータ転送が開始される (④)．
- MPTCP では cwnd（congestion window）の決定に Linked increases algorithm [7] が用いられる．このアルゴリズムでは，すべての subflow の cwnd の合計値 $cwnd_{total}$ はいずれかの subflow で ACK を受け取るたびに増加し，各 subflow の $cwnd_i$ はそれぞれの経路の性能にあわせて $cwnd_{total}$ から割り当てられる．

このように，追加の subflow 2 の確立は subflow 1 の確立のあとに行われるため，それぞれの subflow でのデータ転送の開始には時間差が生じる．第 5 章で述べるように，この時間差が原因となり，TCP 転送の予測手法をマルチパス TCP 転送の予測に適用した場合には予測精度が低下することがある．

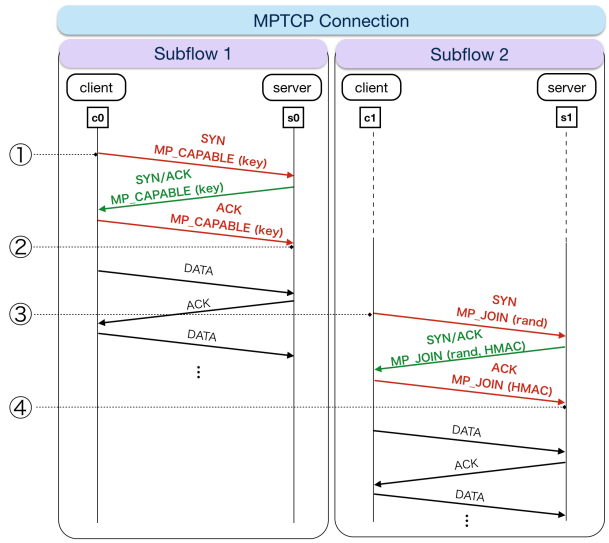

図 2 MPTCP コネクションの確立

4. 実験環境

本研究の実験環境について述べる．第 5 章および第 6 章の実験は Client - Server 間に二つの経路を持つ **図 3** のローカルネットワークで行った．詳細を **表 1** に示す．ただし，Server 上で動作させている Ryu コントローラと OpenFlow switch は，実験用のデータ転送ポートとは異なるポートを通じて接続されている．これらのポートは 表 1 に含めていない．

広域ネットワーク環境を想定した実験を行うため，ネットワークエミュレータ Dummynet [19] を用いて，一部のリンクで遅延制御を行った．もともとの Client - Server 間の RTT は Lagopus switch の転送処理で発生する遅延を含めて 10 ± 2 ms 程度であり，Dummynet を用いて往復 110ms から 170ms の遅延を加えた．

OpenFlow switch には，Lagopus switch [4] を動作させた 8 コア Atom マシンを用いた．Lagopus switch は，汎用サーバで高性能なネットワーク処理を行うための Intel DPDK (Data Plane Development Kit) [1] を用いた OpenFlow 対応のソフトウェアスイッチである．本実験では，8 コア中 7 コアをパケット処理に割り当てた．

OpenFlow コントローラ Ryu [8] には，c0 - s0 が n0 - n1（経路 1）のみを，c1 - s1 が n0 - n2（経路 2）のみを通じて通信するようにタイムアウトなしのフローエントリを事前に登録した．したがって，probe や data の送信中にフロールールに関するコントローラへの問い合わせが発生することはない．

コネクション確立は常に Client から開始し，データの転送も Client から Server へ向けて行う．また，Client が事前に持っている Server のアドレス情報は s0 のみであるとする．したがって，マルチパス TCP 転送時には，経路 1 のコネクション（subflow 1），経路 2 のコネクション（subflow 2）の順で確立が行われる．

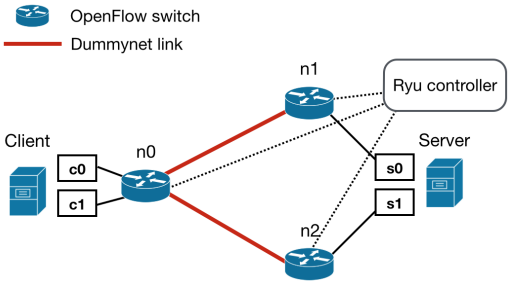

図 3 実験ネットワーク

表 1 実験環境

Client	CPU: Intel Xeon E5-2690 v2 @3.0GHz Memory: 64GB net.ipv4.tcp_wmem max: 200MB
Server	CPU: Intel Xeon E5-2695 v2 @2.4GHz Memory: 192GB net.ipv4.tcp_rmem max: 200MB
Client & Server	OS: Ubuntu 14.04 with Linux kernel 3.14.0 MPTCP: Linux MPTCP kernel 0.89 NIC: Intel I350 2ports
OpenFlow switch	CPU: Intel Atom C2750 @2.4GHz Memory: 16GB OS: Debian 7.8 with Linux kernel 3.2.0 NIC: Intel I210 2ports (n0, n1, n2), Intel X540-AT2 2ports (n0) OpenFlow switch: Lagopus switch 0.11-dev
Dummynet	CPU: Intel Atom C2750 @2.4GHz Memory: 16GB OS: FreeBSD 10.1 NIC: Intel I210 2ports
Link	Ethernet 1000BASE-T
OpenFlow	Protocol: 1.3 Controller: Ryu 3.22 (run on Server)

5. 既存手法による予測

本研究では，まず，単一経路での TCP 転送を想定した既存手法をマルチパス TCP 転送の予測に適用することを検討した．この章では，適用した場合の問題点と予測精度の低下について述べる．

5.1 適用の問題点

第 3 章で述べたように MPTCP は TCP 環境で動作するため，TCP 転送を想定した既存手法をそのまま適用することができる．その場合，probe や data は MPTCP によって複数の subflow に分割して送信され，その送信比率は，輻輳制御などによって各経路の性能に応じて動的に変化する．十分に大きなデータの送信時には各 subflow の送信比率はそれぞれの性能差を反映したものになると考えられるが，probe のようなデータ送信時には subflow 確立の時間差によってそうならない可能性がある．これは，デー

タサイズが小さい場合には，subflow 2 でデータ送信が開始される前に subflow 1 のみで probe の送信が完了し得るためである．その場合，probe では subflow 2 の性能を計測できないため，予測は subflow 1 の性能のみを反映したものになり，実際には subflow 2 も使用されるはずの data 送信時のスループットを正しく予測できないと考えられる．

5.2 各 subflow の送信比率

前節で述べた問題を検証するため，マルチパス TCP 転送時に各経路で送信されるデータ量を計測した．実験は第 4 章の環境において，Dummynet の遅延設定を変更し，3 つの条件下で行った．結果を **図 4** に示す．例として，図中の青の折れ線は，経路 1・経路 2 の RTT がともに 150ms のときの結果を示している．この RTT は，もともとの 10ms に Dummynet による往復 140ms の遅延を加えたものである．横軸は送信したデータサイズ，縦軸は最初に確立される subflow 1 で送信された比率を表している．二つの経路を使用したマルチパス TCP 転送のため，縦軸の値を 100% から減じた値が subflow 2 での送信比率である．

この実験結果から，各 subflow での送信比率は，送信するデータサイズが大きい場合は両経路の性能差をある程度反映するが，データサイズが小さくなるにつれてあまり反映しなくなることがわかる．例えば，経路 1 の RTT が 150ms，経路 2 の RTT が 120ms の場合，256MB のデータは subflow 1 によって 33.9% が送信されている．これは，経路 1 の RTT のほうが大きいため，経路 2 の subflow 2 よりもスループットが低くなることを反映している．しかし，データサイズが 16MB 以下になると徐々に subflow 1 の送信比率が増加していき，64KB の送信時には subflow 2 による送信が全く行われなくなる．つまり，この遅延設定の場合，64KB probe では 256MB data 送信時のスループットに大きく寄与する subflow 2 の性能をまったく計測できないことになる．

スループット予測では，目的の data 送信以外でネットワークリソースを浪費することを避けるために probe サイズは小さいほうが望ましい．しかし，この実験結果から，小さな probe では予測誤差が大きくなると考えられる．

5.3 予測精度の評価

ここでは，既存手法を用いてマルチパス TCP 転送の予測を行った結果を示す．前節の実験結果から，特に小さな probe を用いてマルチパス TCP 転送の予測を行った場合は予測精度が低下することが予想される．そこで，TCP 転送の予測を行った場合の精度とマルチパス TCP 転送の予測に適用した場合の精度の比較を行い，これを確認する．

第 4 章の実験環境で，Dummynet によって各経路の RTT を 120ms から 180ms まで 4ms 刻みで変化させ，そのすべての組み合わせで 2 回ずつ計測を行った．probe サイズを 64KB，data サイズを 256MB として，既存手法を用いてマルチパス TCP 転送と通常の TCP 転送の予測を行った結果を **図 5**，**図 6** に示す．予測モデルの構築には統計ソフト R の 機械学習パッケージ kernlab [3] を利用した．また，予測精度の比較を **表 2** に示す．評価指標には RMSE（Root Mean Square Error）および Fraction of RPE（Relative Prediction Error）[13] を用いた．これらは $\hat{R}$ を予測値，R を実際のスループット，n をデータ数，$\boldsymbol{r}$ をデータセットとして以下のように定義される．

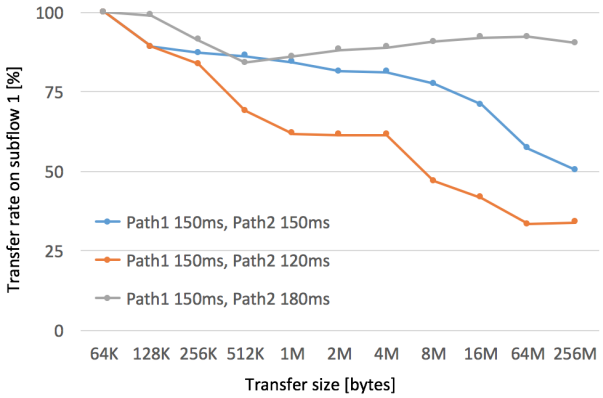

図 4 subflow 1 での送信比率

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{R} - R)^2}$$

$$RPE = \frac{\hat{R} - R}{\min(\hat{R}, R)}$$

$$Fraction\ of\ RPE = \frac{\#\{\boldsymbol{r} | -0.1 < RPE(\boldsymbol{r}) < 0.1\}}{\#\{\boldsymbol{r}\}}$$

図 5 と 図 6 の各データ点は，ある遅延設定において計測した probe と data のスループットの組 (x_i, y_i) を表し，青の直線は得られた回帰予測モデルを表している．これらを比較すると，マルチパス TCP 転送時のデータ点は TCP 転送の予測時と比べて非常に広く分布しており，予測モデルとの乖離が大きいことがわかる．probe と data 送信時のスループットの相関をスピアマンの順位相関係数 ρ を用いて調べると，TCP 転送の予測時は強い相関（$\rho = 0.87$）があるのに対して，マルチパス TCP 転送の予測時は相関が弱い（$\rho = 0.16$）ことがわかった．また，表 2 からは，マルチパス TCP 転送の予測時は RMSE が二倍以上に増加し，Fraction of RPE は 40% ほど低下していることが確認できる．すなわち，予測程度が低下しているといえる．

これらは前節で述べた probe の送信比率の問題から説明できる．例えば，マルチパス TCP 転送によって 64KB probe を送信したとき，その送信のほとんどは subflow 1 のみで行われるため，計測されるスループットもほぼ subflow 1 の性能のみを反映している．しかし data 送信時には subflow 2 も寄与するようになるため，同じ probe スルー

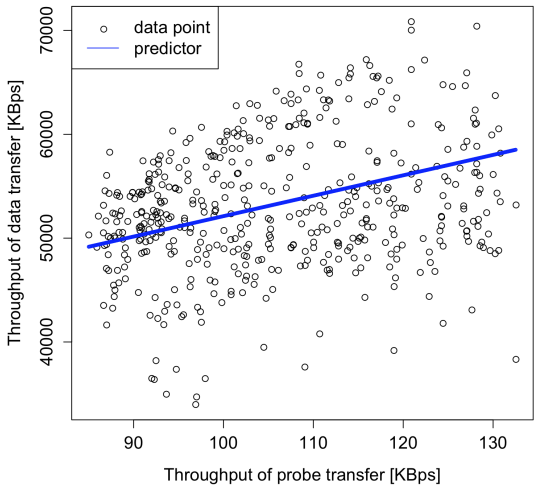

図 5 既存手法によるマルチパス TCP 転送の予測

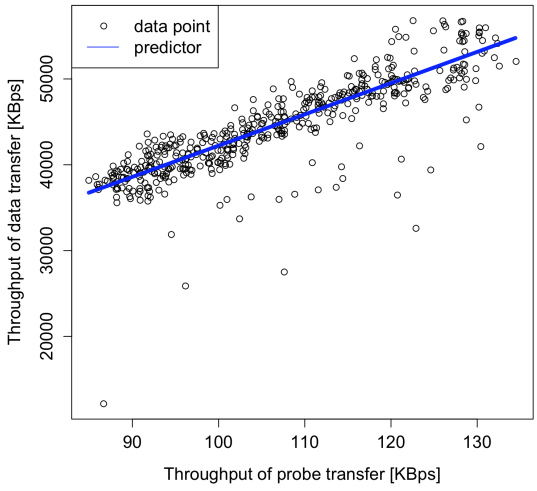

図 6 既存手法による TCP 転送の予測

表 2 既存手法の予測精度: 訓練誤差
(64KB probe，256MB data，訓練データ数 512)

予測対象	RMSE	Fraction of RPE
マルチパス TCP 転送	6126.4 KBps	55.7 %
TCP 転送	3029.9 KBps	94.3 %

プットが計測された場合でも， subflow 2 の性能に応じて data スループットは大きく上下する．そのため，図 5 のような分布が得られ，相関ならびに予測精度の低下が起きる．

6. 提案手法

6.1 概要

提案手法の概要を述べる．第 5 章で述べたように，TCP 転送の予測手法をマルチパス TCP 転送の予測に適用した場合は，probe 送信比率の問題によって特に追加経路の性能を正しく計測できない．その結果，マルチパス TCP 転送の予測では精度が低下する．そこで，既存手法では単一であった probe を経路数に応じて複数用いるように既存手法を拡張した手法を提案する．提案手法概要を以下に示す．

- 訓練データ $(\mathbf{x}_i, y_i)$ $(i = 1, 2, ..., n)$ の組を収集する．
  - マルチパス TCP 転送により data を送信し，スループット y を計測する．
  - マルチパス TCP 転送に使用するすべての経路で，独立した TCP 転送によって probe を送信する．経路 j $(j = 1, 2, ..., m)$ のスループットを x_j として，$\mathbf{x} = (x_1, x_2, ..., x_m)$ を得る．
- 収集した訓練データから，SVR を用いた機械学習によって，入力 $\mathbf{x}_i$ から y_i を求める予測モデル $f : \mathcal{X} \to \mathcal{Y}$ を構築する．
- 予測時には，すべての経路で TCP 転送による probe 送信を行い，$\mathbf{x}$ を得る．$\mathbf{x}$ を f に入力し，出力 y を予測値として得る．

6.2 予測結果

提案手法によってマルチパス TCP 転送のスループット予測を行った結果を示す．Dummynet の遅延設定および収集したデータ数は，第 5.3 節と同様である．data サイズを 256MB，経路あたりの probe サイズを 64KB としたときに得られたデータと予測結果を **図 7** と **図 8** に示す．x 軸と y 軸はそれぞれの経路で TCP 転送によって probe を送信したときのスループット，z 軸はマルチパス TCP 転送によって data を送信したときのスループットを表している．このように複数の probe を用いることで，図 5 では横軸の一次元に集約され，さらに送信比率の偏りによって計測が難しかった各経路の性能を確実に把握することができる．

訓練データに対する予測誤差を **表 3** に示す．各経路で送信した probe サイズは Total probe size の半分である．また，提案手法による精度向上を確認するため，同様の実験環境で行った既存手法による予測の精度を **表 4** に示す．第 5.2 節の結果から，probe サイズを大きくすることで既存手法でも予測精度が向上すると考えられるため，128KB から 128MB の場合について精度を調べた．第 5.2 節で述べたように，32MB や 128MB の probe を用いるのはあまり適切ではないが，これらは比較のために加えた．合計の probe サイズが等しいもの同士を比較すると，提案手法では RMSE は 29.1 % から 26.5 % 程度減少，Fraction of RPE は 30.2 % から 26.4 % 程度増加しており，予測精度が向上していることがわかる．さらに提案手法では，128KB probe による予測でも既存手法で 128MB probe を用いた場合と同等の精度で予測できており，予測コストの観点からも優れていることが確認できた．

提案手法と既存手法の精度の違いを詳しく調べるため，経路 1 の RTT_1 と 経路 2 の RTT_2 の大小関係ごとに Fraction of RPE を求めたものを **表 5** に示す．この結果から，既存手法では RTT_1 が RTT_2 以上のときに精度が低下していることがわかる．これは，第 5.2 節で述べた送信比率の違いによって，よりスループットの高い経路 2 の

表 3 提案手法の予測精度: 訓練誤差
(256MB data, 訓練データ数 512)

Total probe size	RMSE	Fraction of RPE
128KB	4884.8 KBps	77.7%
256KB	4902.9 KBps	77.5 %
512KB	4955.7 KBps	77.5 %

表 4 既存手法の予測精度: 訓練誤差
(256MB data, 訓練データ数 512)

Probe size	RMSE	Fraction of RPE
128KB	6891.8 KBps	47.5 %
256KB	6811.4 KBps	49.6 %
512KB	6740.6 KBps	51.1 %
2MB	6297.5 KBps	53.3 %
8MB	5900.9 KBps	58.2 %
32MB	5234.2 KBps	72.1 %
128MB	4724.1 KBps	75.6 %

表 5 提案手法と既存手法の精度比較: 訓練誤差
(Total 128KB probe, 256MB data, 訓練データ数 512)

各経路の RTT	Fraction of RPE	
	提案手法	既存手法
$RTT_2/RTT_1 > 1.2$	84.2 %	85.5 %
$1.2 \geq RTT_2/RTT_1 > 1.1$	86.6 %	76.1 %
$1.1 \geq RTT_2/RTT_1 > 1$	75.6 %	72.0 %
$RTT_1 = RTT_2$	90.3 %	61.3 %
$1.1 \geq RTT_1/RTT_2 > 1$	89.0 %	28.0 %
$1.2 \geq RTT_1/RTT_2 > 1.1$	86.6 %	25.4 %
$RTT_1/RTT_2 > 1.2$	40.8 %	65.8 %

性能を計測できていないためである．一方，提案手法では複数の probe を用いるため精度が低下しておらず，既存手法よりも高精度である．ただし，RTT_1 が RTT_2 の 1.2 倍よりも大きいときには，既存手法による予測でもあまり精度が低下していない．この原因は不明である．RTT_1 が RTT_2 の 1.2 倍よりも大きいときに提案手法で精度が低下しているのは，該当する遅延設定時の訓練データ数が少ないために，それらのデータに対する誤差を重視した予測モデルが構築されなかったためである．これは，予測モデルのパラメータチューニングや訓練データ数の増加によって改善できると考えられる．

また，収集した 512 データを訓練データと評価データにランダムに分割し，機械学習に使用しない評価データに対する汎化誤差を求めた結果を **表 6** に示す．訓練データ数を減少させたときの精度の変化を調べるため，5 通りの比率で分割し，訓練データとして使用したもの以外はすべて評価データとした．訓練データ数を約 25 にした場合は，未知の評価データに対する Fraction of RPE は 64.4 % に減少しているが，これは，既存手法で 512 データすべてを使用したときの訓練データに対する Fraction of RPE よりも高い値である．

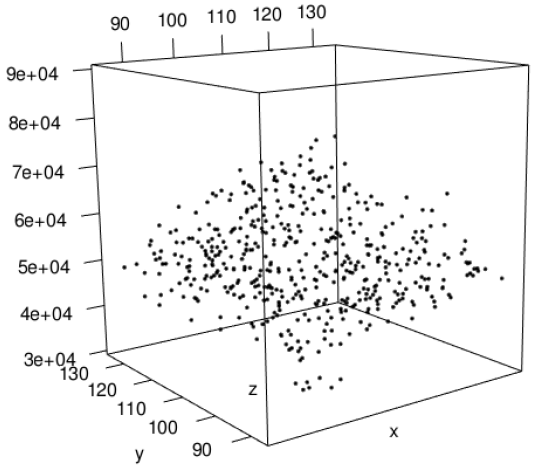

図 7 提案手法により得られたデータの散布図
・: data point
x: Throughput of probe transfer on subflow 1 [KBps]
y: Throughput of probe transfer on subflow 2 [KBps]
z: Throughput of data transfer [KBps]

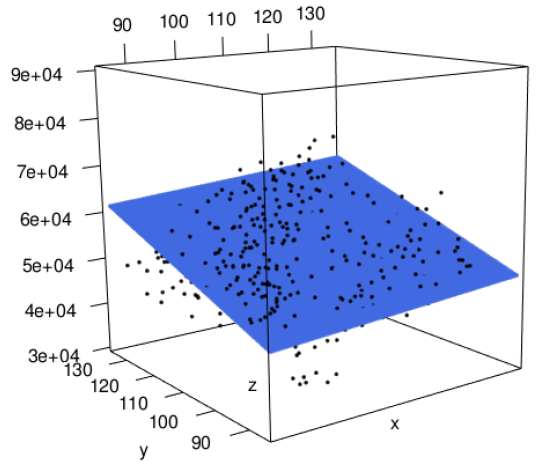

図 8 提案手法により得られた予測モデル
■: predictor

表 6 提案手法の予測精度: 汎化誤差
(Total 128KB probe, 256MB data)

訓練データ数 (% of 512)	RMSE	Fraction of RPE
75 %	3702.2 KBps	78.7 %
50 %	4812.8 KBps	79.1 %
25 %	4946.6 KBps	76.2 %
10 %	5199.1 KBps	73.4 %
5 %	5206.9 KBps	64.4 %

7. おわりに

本稿では，TCP を想定した Lee らの予測手法をマルチパス TCP 転送の予測に適用した結果の報告と，MPTCP の特性を考慮した新たな予測手法の提案を行った．Linux MPTCP kernel を用いた実験によって，既存手法で用いら

れる単一の probe では，MPTCP コネクションの確立方法が原因となり各経路の性能を正しく計測できないことがわかった．そのため，既存手法をそのまま適用した場合は予測精度が低下する．

提案手法は，経路数にあわせて複数の probe を用いることでこの問題に対処した．各経路で独立した TCP 転送によって probe を送信することで，コネクション確立の時間差に影響を受けることなく各経路の性能を正確に計測できる．既存手法との精度比較によって，提案手法では RMSE は最大 29.1 % 減少，Fraction of RPE は最大 30.2 % 増加しており，予測精度が向上していることが確認できた．これは，既存手法で極めて大きな probe を用いた場合と同等の精度であり，予測コストの観点からも優れているといえる．

本研究では，広域ネットワークの再現のため，Dummynet を用いて遅延制御を行った．しかし，これらの設定値は限定的なものであり，クロストラフィックの影響なども調査できていない．そのため，今後は広域テストベッド JGN-X [2] など，より実際の広域ネットワークに近い環境でさらなる評価を行う必要がある．また，提案手法では経路数の増加にともない必要な probe 数も増加するため，予測コストが増大する．そのため，より少ない probe 数で予測できるように probe の送信方法を工夫するなど，手法の改善も行いたいと考えている．

謝辞 本研究は JSPS 科研費 24700031 の助成を受けたものです．

参考文献

[1] DPDK: Data Plane Development Kit. available from ⟨http://dpdk.org/⟩.
[2] JGN-X. 入手先 ⟨http://www.jgn.nict.go.jp/⟩.
[3] kernlab: Kernel-based Machine Learning Lab. available from ⟨http://cran.r-project.org/web/packages/kernlab/index.html⟩.
[4] Lagopus switch: a high performance software OpenFlow 1.3 switch. available from ⟨https://lagopus.github.io/⟩.
[5] Netperf. available from ⟨http://www.netperf.org/netperf/⟩.
[6] Open Networking Foundation. available from ⟨https://www.opennetworking.org/⟩.
[7] RFC 6356: Coupled Congestion Control for Multipath Transport Protocols. available from ⟨https://tools.ietf.org/html/rfc6356⟩.
[8] Ryu SDN Framework. available from ⟨http://osrg.github.io/ryu/⟩.
[9] Multipath TCP (MPTCP). available from ⟨http://datatracker.ietf.org/wg/mptcp/documents/⟩.
[10] 馬場 登志郎.: 日米欧データセンタを活用したデータシステムの遠隔地保管. 入手先 ⟨http://www.ntt.co.jp/journal/0709/files/jn200709021.pdf⟩. NTT 技術ジャーナル（2007 年 9 月）.
[11] 前田 達憲, 阿部 洋丈, 加藤 和彦.: スループット予測による経路選択を用いた OpenFlow 環境での MPTCP 転送. 情報処理学会研究報告, Vol. 2013-OS-127, No.7, 2013 年 12 月.
[12] B. Schölkopf and A.J. Smola and R.C. Williamson and P.L. Bartlet.: *New support vector algorithms* Neural Comput., vol.12, pp.1207-1245, May 2000.
[13] C. Lee, H. Abe, T. Hirotsu and K. Umemura.: *Analytical modeling of network throughput prediction on the internet.* IEICE TRANSACTIONS on Information and Systems, Vol. E95-D, No. 12, pp. 2870-2878, December 2012.
[14] C. Lee, H. Abe, T. Hirotsu and K. Umemura.: *Performance implications of task scheduling by predicting network throughput on the internet.* The 11th IEEE International Symposium on Parallel and Distributed Processing with Applications, Melbourne, Australia, 16-18 July, 2013.
[15] C. Paasch, S. Barre, et al.: Multipath TCP in the Linux Kernel. available from ⟨http://www.multipath-tcp.org/⟩.
[16] C. Raiciu and C. Paasch and S. Barre and A. Ford and M. Honda and F. Duchene and O. Bonaventure and M. Handley.: *How hard can it be? designing and implementing a deployable multipath TCP.* USENIX Symposium of Networked Systems Design and Implementation (NSDI'12), San Jose (CA), 2012.
[17] He, Dovrolis, Ammar.: *On the predictability of large transfer tcp throughput.* In Proc. of ACM SIGCOMM, 2005.
[18] J. Padhye, V.Firoiu, D.Towsley, and J. Kurose.: *Modeling tcp throughput: a simple model and its empirical validation.* IEEE/ACM Transactions on Networking, 8(2):133-145, 2000.
[19] M. Carbone, L. Rizzo.: *Dummynet revisited.* ACM SIGCOMM Computer Communication Review, 40(2) pp. 12-20, March 2010.
[20] M. Mathis, J. Semke, and J. Madhavi.: *The macroscopic behavior of the tcp congestion avoidance algorithm.* ACM Computer Communications Review, 27(3):67-82, 1997
[21] M. Swany, R. Wolski.: *Multivariate resource performance forecasting in the network weather service.* Supercomputing ' 02: Proc. 2002 ACM/IEEE conference on Supercomputing, pp.1-10, Los Alamitos, CA, USA, 2002.
[22] R. Wolski, N. Spring, and J. Hayes.: *The network weather service: A distributed resource performance forecasting service for metacomputing.* J. Future Generation Computing Systems, vol.15, pp.757-768, 1999.