

# 仮想マシンを利用したトランスポートプロトコルの性能評価に関する研究\*

鹿間 敏弘<sup>\*1</sup>

## A Research on Performance Evaluation of Transport Protocols Using Virtual Machines

Toshihiro SHIKAMA<sup>\*1</sup>

<sup>\*1</sup> Faculty of Engineering, Department of Electrical and Electronic Engineering

At present, new transport protocols like MPTCP, SCTP, and CMT-SCTP that can utilize multiple paths over networks have been standardized and implemented on PCs as well as server machines. As there are a large number of parameters having significant effects on the performance of the protocols, a performance evaluation for the cases of various parameters is essential. Although network simulations are traditionally employed for this purpose, available network simulators normally do not cover the new protocols. Another approach is to use real network equipment and PCs, however this requires large equipment costs and labor. This paper proposes performance evaluation using virtual machines on a single PC, where real Linux and FreeBSD operating systems are running on VMware Workstation 14 Player. The new transport protocols are working on these operating systems. It is shown the effectiveness of the proposed approach as well as the comparison of transport protocols by some example experiments.

**Key Words :** Transport Protocols, Multihome, Virtual Machines, MPTCP, SCTP, CMT-SCTP

### 1. 緒 言

情報通信ネットワークのプロトコルの研究開発においてプロトコルの性能評価環境が重要である。従来、このような目的ではネットワークシミュレーションが用いられてきた。これを実現するネットワークシミュレータとしては商用の OPNET<sup>(1)</sup>や QualNet<sup>(2)</sup>など使われているが一般に高価で、学術的にはオープンソースの ns-2<sup>(3)(4)</sup>や ns-3<sup>(5)(6)</sup>が広く使用されてきた。一方、情報通信ネットワークのプロトコルの研究開発が進み、最近ではマルチホーム環境に対応できる MPTCP<sup>(7)-(9)</sup>や SCTP<sup>(10)</sup>および CMT-SCTP<sup>(11)</sup>などのトランスポートプロトコルが使われ始めている。このようなトランスポートプロトコルは複数のネットワーク経路を同時に使用するため、その性能を評価するためには多くのパラメータを評価する必要がある。実機による性能評価が望ましいが、複数のネットワーク遅延シミュレータやパソコン (PC) が必要となり費用や手間のかかる問題がある。また、実験パラメータの設定が複数の機器に分散されるため、実験の自動化が難しい問題もある。ネットワークシミュレーションを利用した評価は、これらの問題を回避できるが、ネットワークシミュレータがマルチホーム対応のトランスポートプロトコルをサポートしていない問題がある。ns-3 のようなネットワークシミュレータへのトランスポートプロトコルの実装には多大な努力が必要で、また実現したとしても現実に使われているプロトコルとの差異が避けられず、評価の信頼性の問題がある。先に述べた商用のネットワークシミュレータも新しいプロトコルのサポートには時間と費用を必要とし、追いついていないのが現状である。このような状況から ns-3 をベースに実際の OS のプロトコルスタックを ns-3 のネットワークシミュレータで動作させる DCE<sup>(12)</sup>が開発されている。DCE は現状 Linux のプロトコルスタックを動作できる段階にあるが、最新バージョンの開発は遅く、また OS も Linux に限られている。

本論文では、上記のようなアプローチとは異なり、仮想マシンに着目して Linux や FreeBSD<sup>(13)</sup>を複数の仮想マシンとして 1 台の物理 PC 上で実行させ、これらの仮想マシン間を仮想的な回線で接続して実験を行う方式を検討する。この方式では、各 OS で実現されているプロトコルスタックを利用できる利点、および 1 台の PC で動作す

\* 原稿受付 2018 年 2 月 23 日

<sup>\*1</sup> 工学部、電気電子工学科

E-mail: shikama@fukui-ut.ac.jp

ため実験パラメータの設定も比較的容易である利点がある。本論文では、最初にマルチホームに対応したトランスポートプロトコルとその性能評価手法について述べ、仮想マシンによる評価環境として VMware<sup>(14)</sup> による手法を説明する。次に、この手法によるマルチホームに対応したトランスポートプロトコルの性能評価例を紹介し、最後にこの手法の特徴と課題についてまとめる。

## 2. マルチホームに対応したトランスポートプロトコルとその性能評価手法

### 2.1 マルチホームに対応したトランスポートプロトコル

有線ネットワークの光化が進み、また携帯電話ネットワークや無線 LAN の普及により、パソコンなどのネットワーク接続は多様化している。最近是有線と無線で同時に複数のネットワークを利用するマルチホームも可能となっている。しかし、従来から使われ長い歴史のあるトランスポートプロトコルである TCP はネットワーク内の単一の経路を利用するように設計されており、複数の経路を同時に利用できない問題がある。詳しく説明すると、下位のネットワーク層は上位 TCP からのセグメントを複数の経路に分散して転送することは可能である。しかし、このような転送を行うとセグメントの到着順序に乱れが生じ、TCP はこの乱れがネットワークの輻輳によりパケットが廃棄されたことによるものか、複数経路への負荷分散により生じたものか区別できない。このため、TCP はセグメントの到着順序が乱れたのは輻輳により中継ルータでパケットが廃棄されたことによるものとみなし、不必要な輻輳制御を行う。この輻輳制御により、TCP のスループットは大幅に低下し、複数経路によるデータ転送の効果を期待できない。

このような TCP の欠点を解決するマルチホーム対応のトランスポートプロトコルとして、MPTCP、SCTP、CMT-SCTP が標準化されている。MPTCP は複数の経路ごとに従来の TCP コネクションによる転送を行い、その上位に複数の TCP コネクションを束ねる機能を追加したもので、Linux などで実装されている。SCTP は TCP とは別に開発されたトランスポートプロトコルで、通信キャリアネットワークの制御信号用に使われている。SCTP はマルチホーム対応ではあるが、同時に複数の経路によるデータ転送は行わない仕様となっている。使用している経路に障害が発生すると、別の経路に直ちに切替えて転送を継続することにより、信頼性の向上を狙ったものである。SCTP は TCP がバイトストリーム転送であるのに対し、ブロック単位でブロックの境界を保存して転送できる特徴、および TCP が単一ストリームの転送しかできないのに対し、複数ストリーム転送ができる特徴がある。CMT-SCTP は SCTP を改良したプロトコルで、複数経路で同時転送できる特徴を有する。Table 1 はトランスポートプロトコルの特徴と利用可能な OS をまとめたものである。機能比較では CMT-SCTP が最も優れていると言えるが、これをサポートしている OS は FreeBSD のみである。

Table 1 Comparison of TCP, MPTCP, SCTP and CMT-SCTP

Transport Protocol	Multi-home	Transfer service	Number of streams	Concurrent Multi-path transfer	OS support
TCP	No	Byte stream	Single	No	All major OSs
MPTCP	Yes	Byte stream	Single	Yes	Linux
SCTP	Yes	Block	Multi	No	Linux, FreeBSD
CMT-SCTP	Yes	Block	Multi	Yes	FreeBSD

### 2.2 性能評価環境

トランスポートプロトコルの性能評価を行う上では、実機による評価とネットワークシミュレータによる評価が従来行われてきた。

#### 2.2.1 実機による評価

実機による評価は、複数の実マシンに評価対象のトランスポートプロトコルを実装した OS をインストールし、実回線により接続するものである。ネットワークの遅延時間や帯域およびパケット損失を疑似的に発生させるため、ネットワーク遅延シミュレータを用いる場合が多い。Fig. 1 は経路数が 2 の場合に対応するための実験構成例を示している。実際に機器を接続するために機器のコストや接続の手間がかかる。Fig.

1 の構成では FreeBSD をインストールした PC を 2 台使用する他に、遅延シミュレータ 2 台、遅延シミュレータの設定を行うための PC が 1 台必要となる。パラメータを変化させて評価実験を行うためには、遅延シミュレータの設定を変化させて行う必要があるが、3 台の PC を連動させることは難しく、実験の自動化が困難な問題がある。

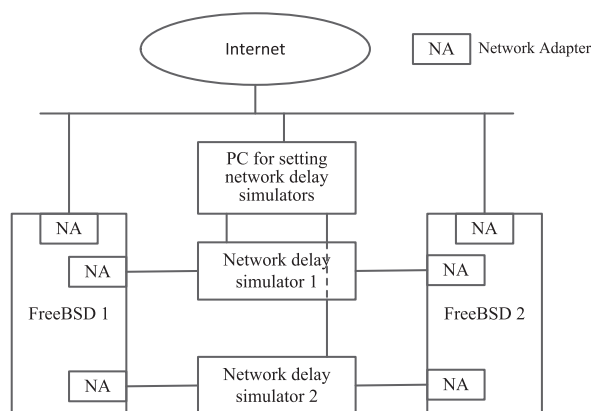


Fig. 1 An example of real network using real devices

### 2.2.2 ネットワークシミュレータによる評価

ネットワークシミュレータは離散事象シミュレーションの原理によりネットワークやプロトコルの評価を行うもので、オープンソースの ns-2 および ns-3 が広く使われている。ネットワークシミュレータを使用すると、スクリプトを用意すればパラメータを変化させた実験を自動的に実行でき、効率良く評価を行える。しかし、ns-2 および ns-3 どちらもマルチホーム環境に対応したトランスポートプロトコルをサポートしていない問題がある。ユーザがプロトコルのプログラムを作成することは可能であるが、一般にマルチホーム環境に対応したトランスポートプロトコルは複雑で、開発は困難である。第三者による ns-2 用および ns-3 用の MPTCP モジュールがインターネットで公開されているが<sup>(15)-(17)</sup>、実際の規格に適合しているか検査が必要で、結果の信頼性に問題がある。

### 2.2.3 DCE による評価

ネットワークシミュレータの欠点を解決する手法として DCE (Direct Code Execution) の開発が進められている。これは ns-3 による離散事象シミュレーション環境で Linux など実際の OS に実装されているプロトコルスタックを動作させるものである。DCE ではネットワーク構成などは ns-3 で記述するが、トランスポートプロトコルは実 OS のものを使用してネットワークシミュレーションを実行するため、信頼性のある結果を期待できるとともに、スクリプトによるシミュレーションの自動実行も容易となる。しかし、現状の DCE はインストールが不安定で動作させるのが難しい問題や、利用可能なプロトコルスタックとして Linux のみの制約がある。

### 2.2.4 仮想マシン環境による評価

本論文では上記のような手法とは異なり、1 台の PC に複数の仮想マシンを実現し、仮想マシン間で通信を行うことによりプロトコルの評価を行う方式を検討する。このような仮想マシン環境を実現するソフトウェアとしては VMware や VirtualBox<sup>(18)</sup> が使われている。最近の PC はマルチコアを有するプロセッサが一般的であり、4 台程度の仮想マシン数であればマルチコアにより動作可能で、複数の仮想マシンを接続したネットワークも可能と考えられる。本論文では Fig. 2 に示すようにフリーで利用できる VMware Workstation 14 Player を Windows 10 上にインストールし、その上に 4 台の仮想マシンを実装した。これら仮想マシン間をネットワーク接続してトランスポートプロトコルの性能評価を行う。Fig. 3 は Fig. 1 に示した実ネットワークと等価な環境を VMware 上の仮想マシンにより構成した例を示している。実ネットワークで必要となったネットワーク遅延シミュレータ相当の機能を VMware により実現できるため、構成が単純となる。Fig. 4 は VMware の仮想マシンの設定メニュー、Fig. 5 は仮想マシンを構成するネットワークアダプタの設定画面を示している。ネットワークアダプタの着信側（受信側）と発信側（送信側）で帯域、パケット損失率、

遅延時間の設定を行うことができる。Fig. 5 では、受信側および送信側の帯域を 10Mbit/s、送信側のパケット損失率を 0.001、送信側の遅延時間を 10ms としている。仮想環境による評価は実際のプロトコルを用いるため、実機による評価に近いが、1 台の PC で実現できるためコスト的および設定の手間の面から有利となる。しかし、全てを 1 台の PC で実現するため大規模なネットワークの実験は行えないためスケーラビリティの問題がある。

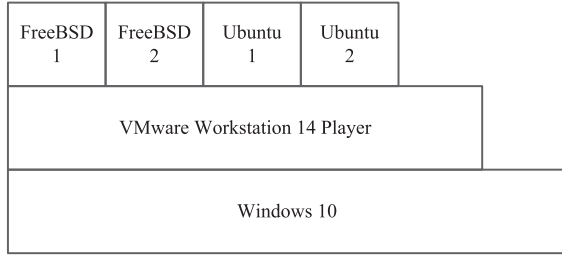


Fig. 2 Structure of virtual machines

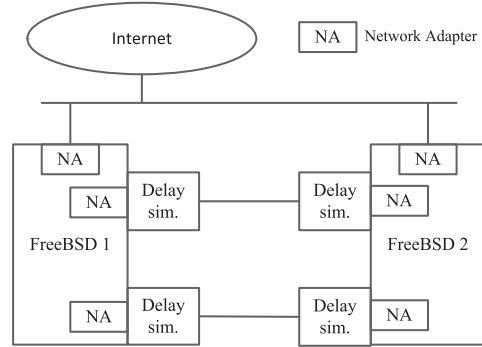


Fig. 3 An example of virtual network using two FreeBSD machines

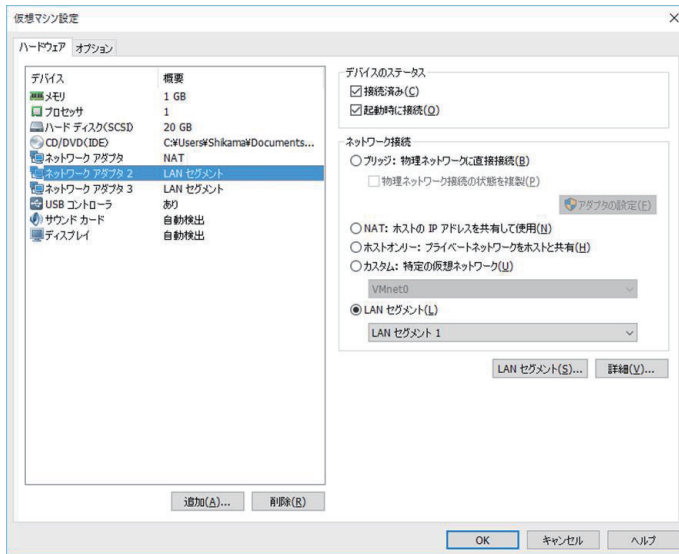


Fig. 4 Setting of virtual machine



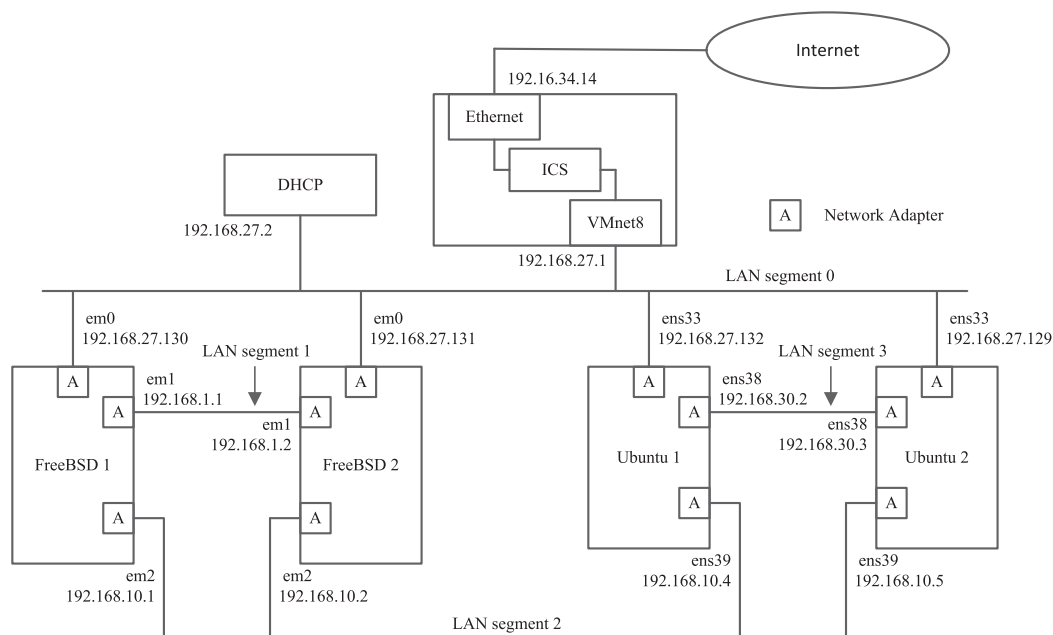
Fig. 5 Setting of network adaptor 2

### 3. マルチホームに対応したトランスポートプロトコル性能評価

#### 3.1 実験に用いた仮想マシンによるネットワーク構成

Fig. 6 は本論文で使用した VMware 上のネットワークの詳細構成を示している。FreeBSD の仮想マシン 2 台と Linux の仮想マシン 2 台を接続している。ここで、Linux のディストリビューションとして Ubuntu を採用した。仮想ネットワーク間の接続は LAN により実現し、FreeBSD の仮想マシンは LAN segment 1 と LAN segment 2, Ubuntu の仮想マシンは LAN segment 3 と LAN segment 2 により接続している。LAN segment 2 はこれら 4 台の仮想マシンを共通に接続している。IP アドレスは固定設定で、LAN segment ごとに 24 ビットのネットワークアドレスを共通にしている。Fig. 6 の上部は VMware および Windows 10 のネットワーク関係の機能を表しており、全 4 台の仮

仮想マシンは LAN segment 0 により Windows を介してインターネットに接続される．このインターネット接続は NAPT<sup>(19)</sup>により Windows の IP アドレスを 4 台の仮想マシンが共用することにより行う．Fig. 6 で Windows 10 の ICS (Internet Connection Sharing) がこの NAPT 機能を実現する．また，仮想マシンのインターネット接続に関し IP アドレスは DHCP により自動割当てされる．



**Fig. 6 Network configuration of the whole system**

使用した PC および OS の仕様を Table 2 に示す．PC は 4 コアかつ 8 スレッドによる実行が可能で，4 台の仮想マシンであれば問題なく動作した．主記憶は 8GB であるが，同時に実行可能な仮想マシン数は主記憶の制限による影響が大きい．ゲスト OS の内，FreeBSD は X window を使用せず，ターミナルベースで使用し，1GB の RAM 割当てで動作した．Ubuntu はウィンドウベースで使用するため，RAM 割当てが 1GB では動作が遅く，1 台は 2GB，他の 1 台は 3GB の RAM を割当てた．

**Table 2 Specifications of PC and software employed**

Specification of PC	Processor	Intel Core i7-3770 CPU @3.4GHz
	No. of cores	4 cores, 8 threads
	Memory	8 GB
	Host OS	Windows 10
Virtualization software	VMware Workstation 14 Player	
Guest OS	Ubuntu 16.04 LTS	
	FreeBSD 11.0	

### 3.2 仮想マシンとネットワークの遅延時間

Fig. 7 は Ubuntu の仮想マシンにおいて VMware で設定した遅延時間と ping コマンドを 100 回送信して測定した平均遅延時間，Fig. 8 はその標準偏差 (Std: Standard deviation) を示している．また，Fig. 9 は FreeBSD の仮想マシンにおいて VMware で設定した遅延時間と ping コマンドにより測定した平均遅延時間，Fig. 10 はその標準偏差を示している．Fig. 7 から分かるように Ubuntu の場合は設定した遅延時間に対し，ほぼ比例して ping コマンドで測定した時間が増加し，標準偏差もほぼ一定である．一方，FreeBSD の場合は，Fig. 9 から分かるように直線的には増加せず，また Fig. 10 のように標準偏差も 20ms の設定の場合に 3.5ms と大きな値となっている．OS の実行周期による影響と推察されるが，Ubuntu の方が精度良く遅延時



間を設定できることが分かる。FreeBSD の場合の遅延時間精度は OS のパラメータ調整で改善できる可能性もあり、今後の課題である。また、VMware を使用せず、FreeBSD では dummynet<sup>(20)</sup>、Ubuntu では TC コマンド<sup>(21)</sup>を用いて帯域、パケット損失、遅延時間の設定可能であるが、FreeBSD の場合は dummynet でも正確な遅延時間が得られない結果となった。

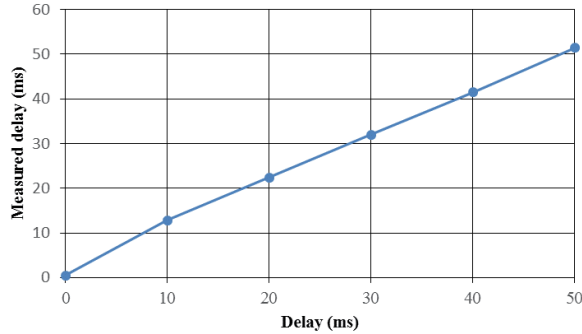


Fig. 7 Average delay measured by ping commands for the case of Ubuntu

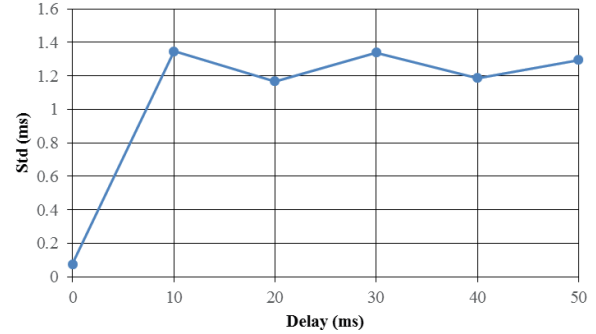


Fig. 8 Standard deviation of delay by ping commands for the case of Ubuntu

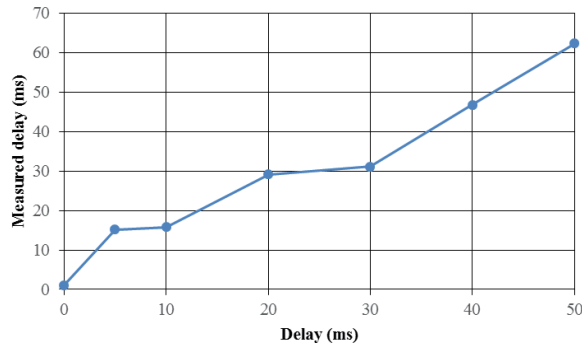


Fig. 9 Average delay measured by ping commands for the case of FreeBSD

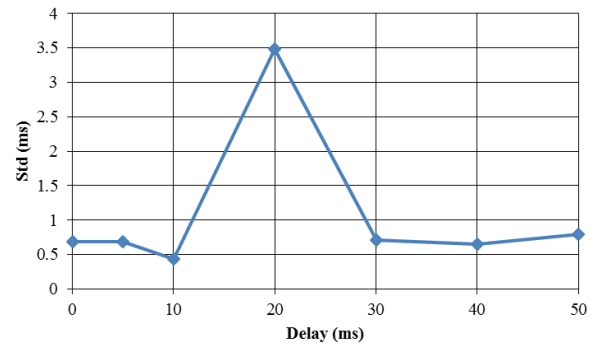


Fig. 10 Standard deviation of delay by ping commands for the case of FreeBSD

### 3.3 MPTCP の性能評価

Fig. 6 の Ubuntu 1 (U1) と Ubuntu 2 (U2) を用いて MPTCP および TCP の性能評価を行った。これら 2 台の仮想マシンを LAN segment 3 と LAN segment 2 で接続することにより、等価的に二つの経路が存在する条件を設定し、Table 3 に示すパラメータで実験を行った。実験では LAN segment 0 を介したインターネットへの接続は遮断した。以下 LAN segment 3 を Path A, LAN segment 2 を Path B と呼ぶ。

Table 3 Parameters for MPTCP and TCP

Path A (LAN segment 3)	Bandwidth	10Mbit/s
	Send delay	15ms
	Receive delay	0ms
	Packet Loss rate: U1 -> U2	0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1
	Packet Loss rate: U2 -> U1	0
Path B (LAN segment 2)	Bandwidth	10Mbit/s
	Send delay	15ms, 50ms
	Receive delay	0ms
	Packet Loss rate: U1 -> U2	0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1
	Packet Loss rate: U2 -> U1	0

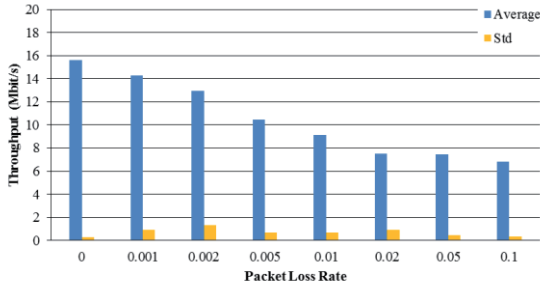
Table 4 は送信側での片側遅延時間設定 (Send delay) に対し, ping コマンドで測定した往復遅延時間を示している. 片側遅延時間設定が 15ms の場合, 平均往復遅延時間は約 35ms, 片側遅延時間設定が 50ms の場合, 平均往復遅延時間は約 104ms の結果となった. この結果は Fig. 7 と整合している.

**Table 4 Measured roundtrip delay for the case of MPTCP**

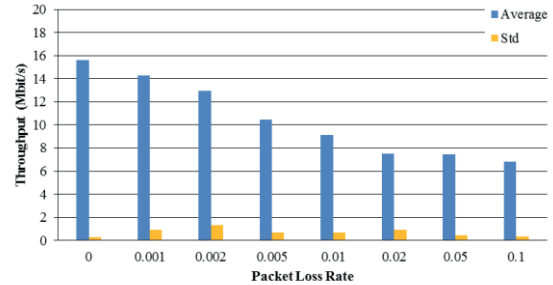
Path (LAN segment)	Send delay (ms)	Average (ms)	Standard deviation (ms)
Path A (LAN segment 3)	15	34.9	1.8
Path B (LAN segment 2)	15	35.1	1.7
Path B (LAN segment 2)	50	104.3	1.8

Fig. 11 は MPTCP のパケット損失率に対するスループット特性で, Path A および B の片側遅延時間を 15ms (平均往復遅延時間は 35ms) とし, Path B のパケット損失率を変化させている. 測定は性能測定ツールである iperf3<sup>(22)</sup>を用いて 12 回行った結果の平均を求め, また標準偏差 (Std) も示している. Fig. 12 も MPTCP のパケット損失率に対するスループット特性で, Path B の片側遅延時間を 50ms (平均往復遅延時間は 104ms) とし, Path B のパケット損失率を 0, Path A のパケット損失率のみを変化させている. Fig. 13 は Fig. 12 と同じであるが, Path A のパケット損失率を 0, Path B のパケット損失率のみを変化させている.

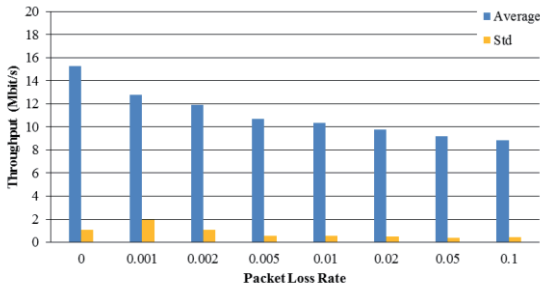
パケット損失率が 0 の場合, Fig. 11 の場合では約 18 Mbit/s, Fig. 12 と Fig. 13 の場合は 15.5 Mbit/s 程度のスループットが得られ, 一本の Path の帯域である 10 Mbit/s より大きい値となっており, 二本の Path を束ねて有効利用していることが分かる. Fig. 12 では片側遅延時間が 15ms の Path A でパケット損失が発生し, Fig. 13 では片側遅延時間が 50ms の Path B でパケット損失が発生する. パケット損失率が 0.01 以上の場合, Fig. 13 の方が大きなスループットが得られている. これはパケット損失が発生しない方の Path においてスループットが大きいためである. Fig. 14 は Fig. 11, Fig. 12, Fig. 13 の結果をまとめたもので, それぞれ Case 1, Case 2, Case 3 として示している.



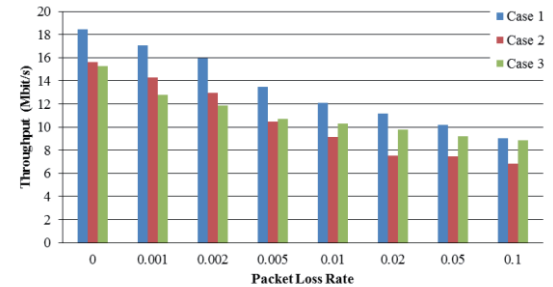
**Fig. 11 MPTCP throughput vs. packet loss rate for the case where delay of path A and B is 15ms and packet loss occurs on path B**



**Fig. 12 MPTCP throughput vs. packet loss rate for the case where delay of path B is 50ms and packet loss occurs on path A**



**Fig. 13 MPTCP throughput vs. packet loss rate for the case where delay of path B is 50ms and packet loss occurs on path B**



**Fig. 14 Summary of MPTCP throughput vs. packet loss rate**

Fig. 15 は Path A で TCP を使用した場合のスループット, Fig. 16 は Path B で片側遅延時間を 50ms とした場合の TCP のスループットを示している. Fig. 15 の場合の平均往復遅延時間は 35ms, Fig. 16 の場合の平均往復遅延時間は 104ms で, 平均往復遅延時間が小さい方がパケット損失率の増加に対してスループットの低下が小さいことが分かる. これはパケット損失により引き起こされる輻輳制御により輻輳ウィンドウの大きさが制限されるため, 平均往復遅延時間が小さい方が大きなスループットが得られることによる.

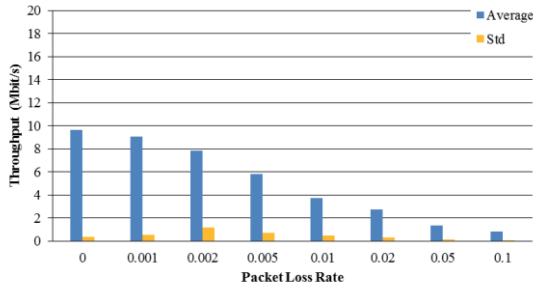


Fig. 15 TCP Throughput vs. Packet Loss Rate on path A

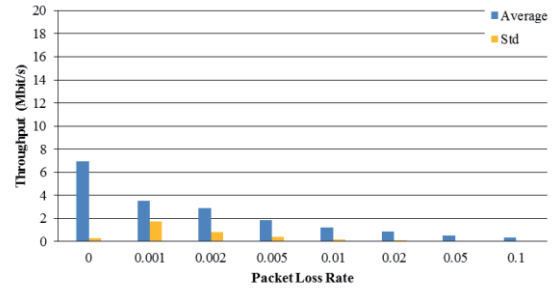


Fig. 16 TCP Throughput vs. Packet Loss Rate on path B where delay of path B is 50ms

### 3.4 CMT-SCTP の性能評価

Fig. 6 の FreeBSD 1 (BSD1) と FreeBSD 2 (BSD2) を用いて CMT-SCTP および SCTP の性能評価を行った. これら 2 台の仮想マシンを LAN segment 1 と LAN segment 2 で接続し, MPTCP および TCP の場合と同様に等価的に二つの経路が存在する条件を設定し, Table 5 に示すパラメータで実験を行った. この場合も LAN segment 0 を介したインターネットへの接続は遮断した. 以下 LAN segment 3 を Path C, LAN segment 2 を Path B と呼ぶ.

Table 5 Parameters for CMT-SCTP and SCTP

Path C (LAN segment 1)	Bandwidth	10Mbit/s
	Send delay	10ms
	Receive delay	0ms
	Packet Loss rate: BSD1 -> BSD2	0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1
	Packet Loss rate: BSD2 -> BSD1	0
Path B (LAN segment 2)	Bandwidth	10Mbit/s
	Send delay	10ms, 50ms
	Receive delay	0ms
	Packet Loss rate: BSD1 -> BSD2	0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1
	Packet Loss rate: BSD2 -> BSD1	0

Table 6 は送信側での片側遅延時間 (Send delay) に対し, ping コマンドで測定した平均往復遅延時間を示している. 片側遅延時間が 10ms の場合, Path A の平均往復遅延時間は約 26ms, Path B の平均往復遅延時間は約 30ms, 片側遅延時間が 50ms の場合, 平均往復遅延時間は約 121ms の結果となった. Table 5 の Ubuntu による MPTCP の結果に比べ, 設定誤差が大きくまた標準偏差からバラツキが大きいことも分かる. Path A と Path B で同じ片側遅延時間 10ms に設定した場合でも, 平均往復遅延時間は異なっている.

Table 6 Measured roundtrip delay for the case of CMT-SCTP

Path (LAN segment)	Send delay (ms)	Average (ms)	Standard deviation (ms)
Path C (LAN segment 1)	10	25.9	3.3
Path B (LAN segment 2)	10	30.0	2.9
Path B (LAN segment 2)	50	121.2	8.8



Fig. 17 は CMT-SCTP のパケット損失率に対するスループット特性で、Path C および B の片側遅延時間を 10ms（平均往復遅延時間は 26ms と 30ms）とし、Path C のパケット損失率を変化させている。測定は MPTCP および TCP の場合と同様に性能測定ツールである iperf3 を用いて行った。Fig. 18 も CMT-SCTP のパケット損失率に対するスループット特性で、Path B の片側遅延時間を 50ms（平均往復遅延時間は 121ms）とし、Path B のパケット損失率を 0 とし、Path C のパケット損失率のみを変化させている。Fig. 19 は Fig. 18 と同じであるが、Path C のパケット損失率を 0 とし、Path B のパケット損失率のみを変化させている。

パケット損失率が 0 の場合、Path C および B の片側遅延時間を 10ms とした Fig. 17 では約 14.5 Mbit/s 程度のスループットが得られ、一本の Path の帯域である 10 Mbit/s よりも大きい値となっており、二本の Path が有効使用されていることが分かる。しかし、MPTCP はパケット損失率が 0 の場合、約 18 Mbit/s を超えるスループットが得られており、CMT-SCTP より性能が優れていると言える。これは CMT-SCTP がマルチストリームに対応するため、その分ヘッダなどのオーバーヘッドが大きいことによるものと考えられる。Fig. 20 は Fig. 17, Fig. 18, Fig. 19 の結果をまとめたもので、それぞれ Case 4, Case 5, Case 6 として示している。

Path C の片側遅延時間を 10ms（平均往復遅延時間は 26ms）、Path B の片側遅延時間を 50ms（平均往復遅延時間は 121ms）とした Fig. 18 と Fig. 19 では、パケット損失率が 0 でもスループットが 8 Mbit/s 程度となっており、二本の Path を束ねて有効利用しているとは言えない結果となった。原因として CMT-SCTP の場合、受信側のバッファ不足によるものと推測されるが、原因究明は今後の課題である。

二つの Path の遅延時間が異なる場合、受信側ではセグメントの到着順序が乱れ、これを正しい順序に戻すために大きな受信バッファが必要となる<sup>(23)</sup>。Fig. 21 はこの受信側での順序制御（Resequencing）を示している。MPTCP の場合、同じ状況でも大きなスループットが得られたが、Ubuntu の Linux カーネルでは受信バッファの自動調整を行っており<sup>(24)</sup>、これにより大きな受信バッファが動的に割り当てられ、スループットが向上している可能性がある。この検証は今後の課題である。

Fig. 22 は Path C で SCTP を使用した場合のスループット、Fig. 23 は Path B で片側遅延時間を 50ms とした場合の SCTP のスループットを示している。Fig. 22 の場合の平均往復遅延時間は 26ms、Fig. 23 の場合の平均往復遅延時間は 121ms で、平均往復遅延時間が小さい方がパケット損失率の増加に対してスループットの低下が小さい。これは TCP の場合と同様に、パケット損失により引き起こされる輻輳制御により輻輳ウィンドウの大きさが制限されるため、遅延時間が小さい方が大きなスループットが得られることによる。

Fig. 24 および Fig. 25 は Fig. 22 および Fig. 23 と同じ条件でトランスポートプロトコルを TCP に変更して測定した結果を示している。Fig. 22 および Fig. 23 とほぼ同じ傾向の結果が得られているが、例えばパケット損失率が 0 の場合、Fig. 22 と Fig. 24 を比較すると、TCP の方が 10% 程度大きなスループットの結果となっており、SCTP のヘッダのオーバーヘッドが大きいことによる影響と考えられる。また、Fig. 15 および Fig. 16 に示した Ubuntu の場合の TCP のスループットと比較すると、Ubuntu のスループットの方が若干大きく、受信バッファ量の差異による影響と考えられる。

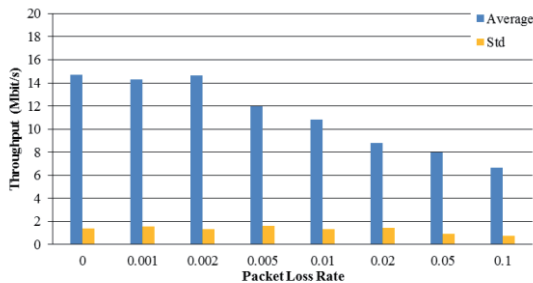


Fig. 17 CMT-SCTP throughput vs. packet loss Rate for the case where delay of path C and B is 10ms and packet loss occurs on path C

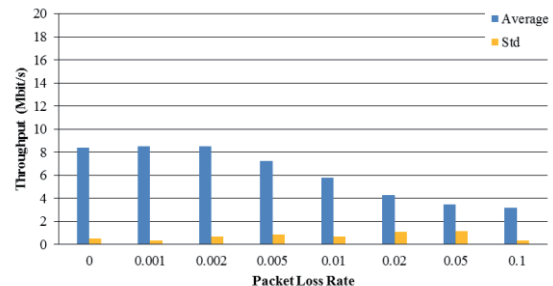


Fig. 18 CMT-SCTP throughput vs. packet loss rate for the case where delay of path B is 50ms and packet loss occurs on path C

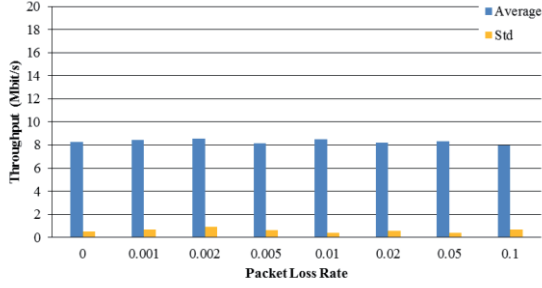


Fig. 19 CMT-SCTP throughput vs. packet loss rate for the case where delay of path B is 50ms and packet loss occurs on path B

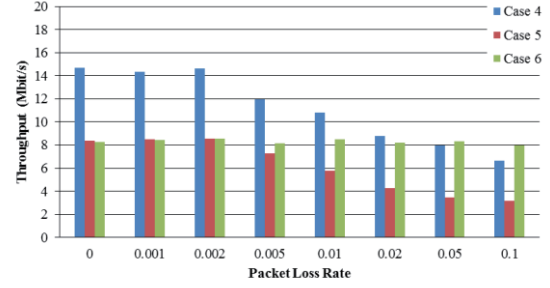


Fig. 20 Summary of CMT-SCTP throughput vs. packet loss rate

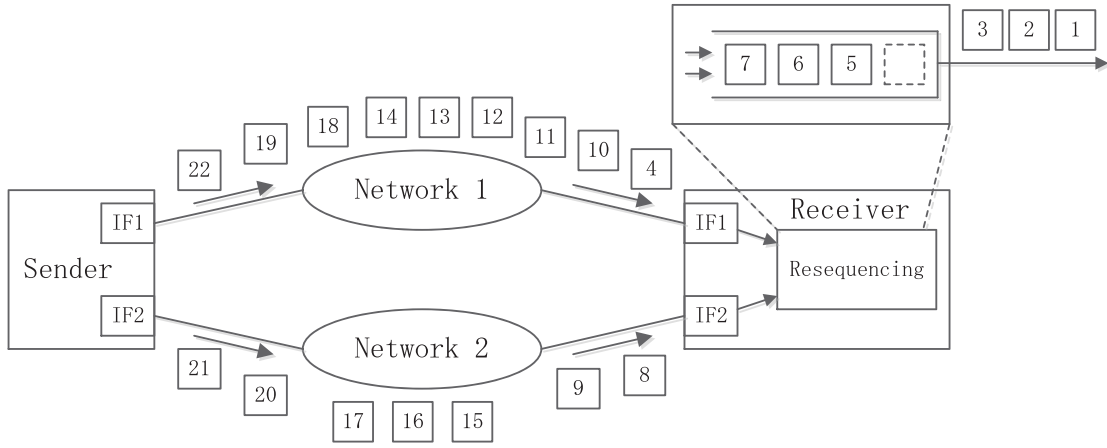


Fig. 21 Resequencing by the receiving side of transport protocol

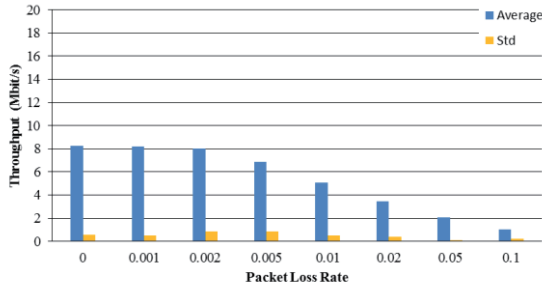


Fig. 22 SCTP throughput vs. packet loss rate on path C

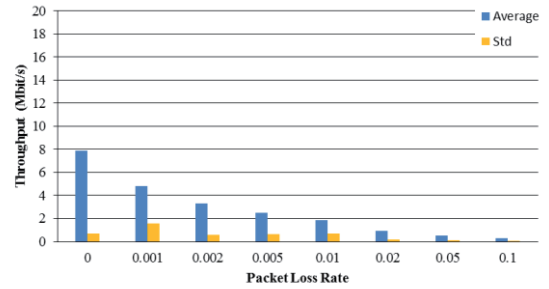


Fig. 23 SCTP throughput vs. packet loss rate on path B where delay is 50 ms

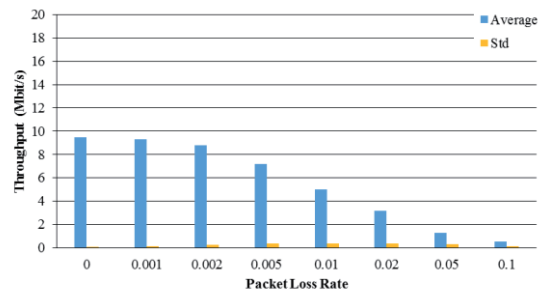


Fig. 24 TCP throughput vs. packet loss rate on path C

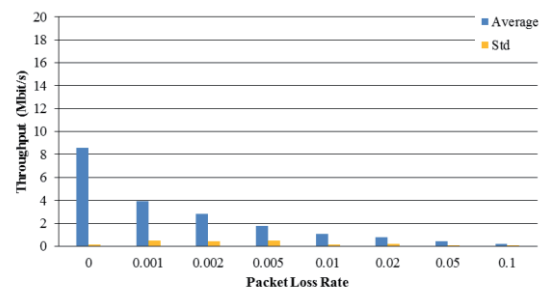


Fig. 25 TCP throughput vs. packet loss rate on path B where delay is 50 ms

## 4. 結 言

本論文では、マルチホームに対応するトランスポートプロトコルの性能評価方式について検討し、仮想マシンによるトランスポートプロトコルの性能評価が有効であることを示した。FreeBSD と Ubuntu を仮想マシンとして動作させた結果、FreeBSD の場合の遅延時間設定の精度不良が判明した。今後、原因の追究と対策が必要である。一つの方法として、FreeBSD による仮想マシンの間で、Ubuntu による仮想マシンをルータとして動作させてパケットの中継を行わせ、精度の良い Ubuntu で遅延時間の設定を行うことにより問題を回避できると考えられるが、構成が複雑となり今後の課題である。

本論文では Ubuntu による仮想マシンで MPTCP, FreeBSD による仮想マシンで CMT-SCTP の性能評価を行った。MPTCP では二つの経路の遅延時間差がある場合でも比較的良好なスループットが得られたが、CMT-SCTP では遅延時間差がある場合に二つの経路の帯域を有効利用できる結果とはならなかった。これはプロトコルの性能の問題よりも Ubuntu と FreeBSD における実装方式の差による影響が大きいと考えられる。また、二つの経路の遅延時間が同じ場合でも MPTCP のスループットの方が大きい結果となった。これは CMT-SCTP がマルチストリーム対応のためその分のヘッダのオーバーヘッドが大きいと考えられる。Ubuntu と FreeBSD で受信バッファ管理の差異による影響については、今後の検証が必要である。

本論文では、Ubuntu の MPTCP と FreeBSD の CMT-SCTP の比較を行ったが、Ubuntu では SCTP も使用可能である。Ubuntu の SCTP と FreeBSD の SCTP の比較が必要であるが、性能評価ツール iperf3 が Ubuntu の SCTP に関しては不具合により動作しなかった。この不具合が解消されたら評価を行う予定である。

## 付 録

Ubuntu における MPTCP および FreeBSD における CMT-SCTP は、独立したソフトウェアではなく、それぞれ TCP および SCTP の中に組込まれ、既存のソケットインタフェースを介して利用でき、上位のソフトウェアに対して互換性を保証している。これらの拡張プロトコルはデフォルトで無効となっており、以下のような設定により有効にできる。

Ubuntu による MPTCP の場合は、以下のファイルに 0 または 1 を書き込むことにより、MPTCP のマルチホーム機能を OFF または ON にできる。

```
/proc/sys/net/mptcp/mptcp_enabled
```

FreeBSD による CMT-SCTP の場合は、以下のように設定を行うとマルチホーム機能が有効となる。

```
sysctl net.inet.sctp.cmt_on_off=1
```

FreeBSD による CMT-SCTP の設定状態は、以下により把握できる。

```
sysctl net.inet.sctp.cmt_on_off
```

## 文 献

- (1) 情報工房, “OPNET Modeler”, [http://www.johokobo.co.jp/opnet\\_rd/opnet\\_rd\\_index.html](http://www.johokobo.co.jp/opnet_rd/opnet_rd_index.html) (参照日 2018 年 2 月 18 日).
- (2) 構造計画研究所, “QualNet”, <http://network.kke.co.jp/products/qualnet/> (参照日 2018 年 2 月 18 日).
- (3) VINT project, “The Network Simulator - ns-2”, <http://www.isi.edu/nsnam/ns/> (参照日 2018 年 2 月 22 日).
- (4) 水野秀樹, NS2 によるネットワークシミュレーション入門 - 有線からワイヤレスアドホックネットワークまで, 第 1 版 (2011), 森北出版.
- (5) “ns-3”, <https://www.nsnam.org/> (参照日 2018 年 2 月 22 日).
- (6) 銭 飛, ns3 によるネットワークシミュレーション, 第 1 版 (2014), 森北出版.
- (7) A. Ford, C. Raiciu, M. Handley and O. Bonaventure, “TCP Extensions for Multipath Operation with Multiple Addresses”, RFC6824 (2013).

- (8) A. Ford, C. Raiciu, M. Handley, S. Barre and J. Iyengar, “Architectural Guidelines for Multipath TCP Development”, RFC6182 (2011).
- (9) Y. Nishida, “Multipath TCP の紹介と最近の動向”, [https://www.isoc.jp/materials/20131220/20131220\\_mptcp.pdf](https://www.isoc.jp/materials/20131220/20131220_mptcp.pdf) (参照日 2018 年 2 月 18 日).
- (10) R. Stewart, “Stream Control Transmission Protocol”, RFC4960 (2007).
- (11) P. Amer, M. Becke, T. Dreibholz, N. Ekiz, J. Iyengar, P. Natarajan, R. Stewart and M. Tuexen, “Load Sharing for the Stream Control Transmission Protocol (SCTP)”, draft-tuexen-tsvwg-sctp-multipath-15.txt (2018).
- (12) “DIRECT CODE EXECUTION”, <https://www.nsnam.org/overview/projects/direct-code-execution/> (参照日 2018 年 2 月 18 日).
- (13) “FreeBSD プロジェクト”, <https://www.freebsd.org/ja/> (参照日 2018 年 2 月 18 日).
- (14) VMware, “VMware”, <https://www.vmware.com/jp.html> (参照日 2018 年 2 月 18 日).
- (15) Y. Nishida, “MPTCP implementation on ns-2”, <http://www.jp.nishida.org/mptcp/> (参照日 2018 年 2 月 19 日).
- (16) M. Kheirkhah, I. Wakeman and G. Parisis, “Multipath-TCP in ns-3”, [http://blog.multipath-tcp.org/blog/html/\\_downloads/mptcp-bib.pdf](http://blog.multipath-tcp.org/blog/html/_downloads/mptcp-bib.pdf) (参照日 2018 年 2 月 19 日).
- (17) M. Kheirkhah, “MultiPath TCP (MPTCP) Implementation in ns-3”, <https://github.com/mkheirkhah/mptcp> (参照日 2018 年 2 月 19 日).
- (18) Oracle, “VirtualBox”, <https://www.virtualbox.org/> (参照日 2018 年 2 月 18 日).
- (19) 松山公保, 基礎からわかる TCP/IP ネットワークコンピューティング入門, 第 3 版 (2015), p. 279, オーム社.
- (20) L. Rizzo, “The dummynet project”, <http://info.iet.unipi.it/~luigi/dummynet/> (参照日 2018 年 2 月 19 日).
- (21) Linux Foundation Wiki, “netem”, <https://wiki.linuxfoundation.org/networking/netem> (参照日 2018 年 2 月 18 日).
- (22) ESnet, “iperf3”, <https://github.com/esnet/iperf> (参照日 2018 年 2 月 20 日).
- (23) 鹿間敏弘, “マルチパス TCP の順序制御に関する性能評価”, 福井工業大学研究紀要, 第 46 号 (2016), pp. 8-17.
- (24) 広瀬大志, 鹿間敏弘, “MPTCP の ns-3 と Linux によるシミュレーションの比較”, 電子情報通信学会 2017 年総合大会 通信講演論文集 2 (2017), p.65.

(平成 30 年 3 月 31 日受理)