マルチパス TCP の順序制御に関する性能評価*

鹿間 敏弘[*1]

Performance Evaluation of Resequencing by Multipath TCP

Toshihiro SHIKAMA[*1]

[*1] Faculty of Engineering, Department of Electrical and Electronic Engineering

Multi-home environment, where a terminal is connected to a number of networks through multiple interfaces, is prevailing. For such a situation, IETF has standardized Multipath TCP (MPTCP), which has been introduced into commercially available operating systems. While conventional TCP can utilize only a single path, MPTCP can utilize multiple paths simultaneously and improve throughput as well as reliability. Since MPTCP segments application data into a number of blocks and distributes them over multiple paths, the receiving side of MPTCP has to perform resequencing function which reassembles received blocks to recover original data. This resequencing requires a large number of buffers to attain high throughput. This paper investigates the performance of MPTCP from the view point of the resequencing function and associated volume of buffers by using network simulator ns-2.

Key Words : Multipath TCP, MPTCP, Resequencing, ns-2, Network Simulations

1. 緒　　言

今日，光ファイバによる有線通信に加え無線通信が広く使われ，通信手段が多様化している．これまで，端末は一つのネットワークに接続する利用形態が主であったが，スマートフォンなどは 3G や LTE などの携帯電話回線とともにローカルな WiFi など，独立した複数ネットワークを同時に利用可能である．インターネットでは，このように端末が複数インタフェースを介してネットワーク利用できる環境をマルチホームと呼ぶ(1)．過去には特殊ケースとされてきたマルチホームであるが，最近は一般的になりつつある．

このようなマルチホームに対し，従来広く使われてきた TCP は同時に 1 経路（ネットワーク）しか使用できず，複数ネットワークの同時利用はできなかった．しかし最近，複数経路利用可能なマルチパス TCP（以下，MPTCP と略す）の標準化が進み，iOS や Linux などで利用可能となっている(2)(3)．Fig. 1 に MPTCP が適用されるネットワークの利用構成例を示す．同時に複数のネットワークを利用してデータを分散して転送することにより通信帯域を拡大できる．また，障害により一つの経路が使用不可となっても，残りの経路で通信を継続することにより通信の信頼性を向上させることが期待できる．

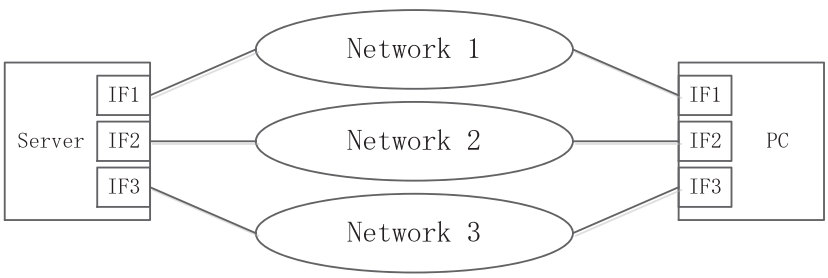

Fig. 1 An example of multi-home networking environment where MPTCP is applied

* 原稿受付　2016 年 2 月 24 日
[*1] 工学部　電気電子工学科
E-mail: shikama@fukui-ut.ac.jp

従来，マルチホームに対応できるトランスポートプロトコルとして SCTP[4]が IETF により標準化され，通信キャリアのネットワークなどで既に使用されている．しかし，この SCTP には次のような欠点がある．

- マルチホームによる複数の経路を同時には利用できず，ある時点ではどれか一つを利用する．ただし，障害などで経路が使用不可となった場合は，別の経路に切り替えて使用する．
- SCTP が提供するアプリケーションとのインタフェースは独自となっており，TCP で提供されるソケットインタフェースを利用するアプリケーションプロラムは利用できない．

本論文で評価を行う MPTCP ではこのような SCTP の問題点が解決され，使い易く導入が容易となっている．MPTCP は近年広く使われているスマートフォンにおいて活用されている．スマートフォンは LTE と WiFi のように複数のインタフェースでネットワークに接続でき，マルチホーミングなネットワーク環境となっている．iphone では iOS により，MPTCP が利用されており，広く使われているアプリケーション Siri が MPTCP を利用していることが知られている[5]．

MPTCP を活用すると，企業などの組織内で複数拠点間をインターネットで結ぶイントラネットの信頼性を向上できると考えられる．Fig. 2 は TCP プロキシに MPTCP を利用したイントラネットの構成例を示している．TCP プロキシは端末やサーバからの TCP コネクションを終端し，インターネットには別の TCP コネクションを設定して，これらコネクション間の中継を行うものである[6]．主にセキュリティ対策および無線通信を含むネットワークにける性能向上手段として広く使用されている．プロキシ間の接続に MPTCP を用いると，PC やサーバには MPTCP によるデータ転送能力が無くても，TCP プロキシが MPTCP をサポートすることにより，拠点間では MPTCP による通信が可能となる．これにより拠点間で複数のネットワークを利用でき，通信帯域を増加させるとともにネットワークの信頼性を向上できる．

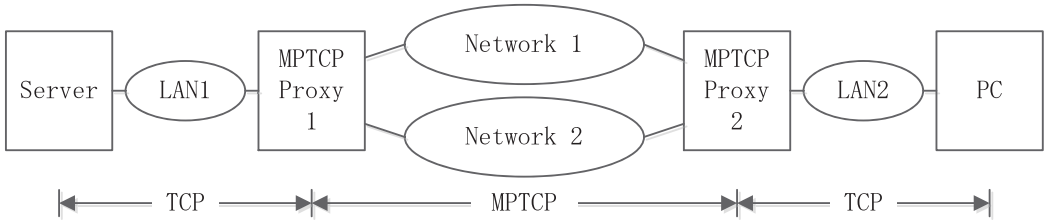

Fig. 2 An example of network configuration using MPTCP proxies

MPTCP のプロトコルは IETF (The Internet Engineering Task Force) により RFC6824[7]および RFC6812[8]として標準化され，製品への実装も進んでいる．しかし，MPTCP の性能に関して十分研究されているわけではない．MPTCP では複数のネットワークにデータを分配するが，受信側ではデータをもとに戻す順序制御が必要である．順序制御には受信側で受信バッファを利用することにより行うことが，通信帯域や遅延時間そして通信品質の異なるネットワークを利用する場合，条件によっては大量の受信バッファが必要となる．本件では，この受信側の順序制御に注目して MPTCP の性能を明らかにすることを目的としている．

以下，本論文では第 2 章で MPTCP の概要と研究課題を述べ，第 3 章では，ネットワークシミュレータ ns-2 による MPTCP のシミュレーションを説明する．第 4 章ではシミュレーションによる評価結果を述べ，第 5 章で結論を述べる．

2. MPTCP の概要と本研究の目的

2.1 MPTCP の概要

MPTCP のレイヤ構成を Fig. 3 に示す[9]. MPTCP は従来の TCP の上位かつアプリケーション層の下位に位置し，既存の TCP による複数のコネクション（以下，サブフローと言う）にデータを分配し，受信したデータをもとのデータに戻す順序制御を行う．Fig. 3 では subflow1 から subflow3 までの 3 本のサブフローが存在する場合を示している．MPTCP と上位のアプリケーション間はソケットインタフェースが用いられ，既存のアプリケーションは変更することなく MPTCP を利用できる特徴がある．

MPTCP のプロトコル情報は TCP のオプションフィールドを拡張して定義されている．各 TCP のサブフローでは TCP で定義されたシーケンス番号が用いられるが，これに加え複数経路に分配されるデータの順序を保存するため，MPTCP 全体で独立したバイト単位のシーケンス番号が用いられ，これにより受信側での順序制御が行われる．MPTCP は 2 種類のシーケンス番号の対応関係を把握しておく必要があり，このためにテーブルが使用される．

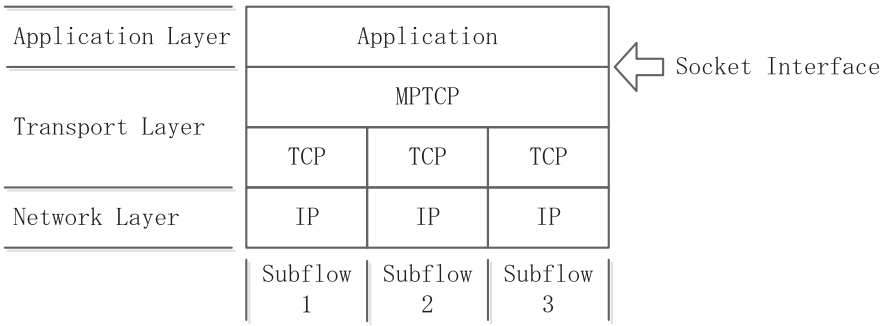

Fig. 3 Layer structure of MPTCP

2.2 MPTCP における順序制御

MPTCP では受信側で複数経路に分配されたデータを組み立てる順序制御が必須となる．Fig. 4 は MPTCP における順序制御を示している．通信経路ごとに遅延時間が異なるため，受信側ではデータを受信バッファに保留して元の順序に並べ替えることが必要で，バッファ用のメモリが必要となる．Fig. 4 で，番号 1，2，3 のデータは正しい順序で受信されて上位レイヤに渡され，番号 5，6，7 のデータは番号 4 のデータより先に受信された場合を示している．番号 5，6，7 のデータは番号 4 のデータが受信されるまで，バッファに保留される．このように，下位の TCP によるデータ転送が終了しても，上位の MPTCP 層が順序の整合しないデータを保留するため，順序制御による追加の遅延時間が生じる．

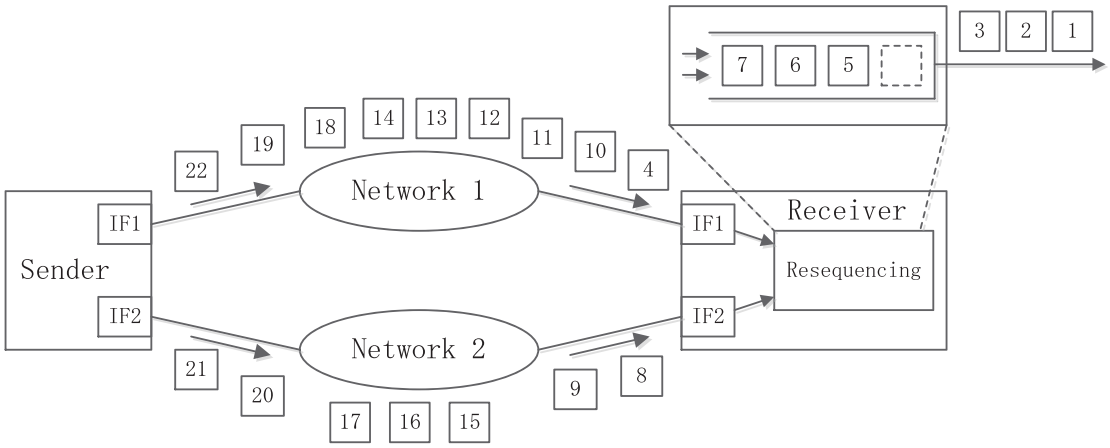

Fig. 4 Resequencing by the receiving side of MPTCP

2.3 MPTCP における順序制御

順序制御は遅れて到着したデータが受信されるとこれを待ってバッファに保留していたデータを一斉に解放するため，遅延時間を発生させるとともにデータ出力がバースト的となる問題を引き起こす．Fig. 5 は Fig. 4 の場合におけるバースト的な出力を時系列的に示している．Fig. 5 において番号 4 のデータが受信されると，番号 4，5，6，7 のデータが一斉に上位層に出力され，バースト的な出力が発生することが分かる．Fig. 5 ではバースト的に出力されるデータ数が 4 程度であるが，番号 4 のデータがより大きく遅れて到着すると，バースト的に出力はさらに大きくなる．高速の通信を行う場合，数 100 から数 1000 セグメ

ント分のデータが一斉に出力される場合もあり得る. このようなバースト的な出力は Fig. 2 のような MPTCP を TCP プロキシとして利用する場合に問題となる可能性がある.

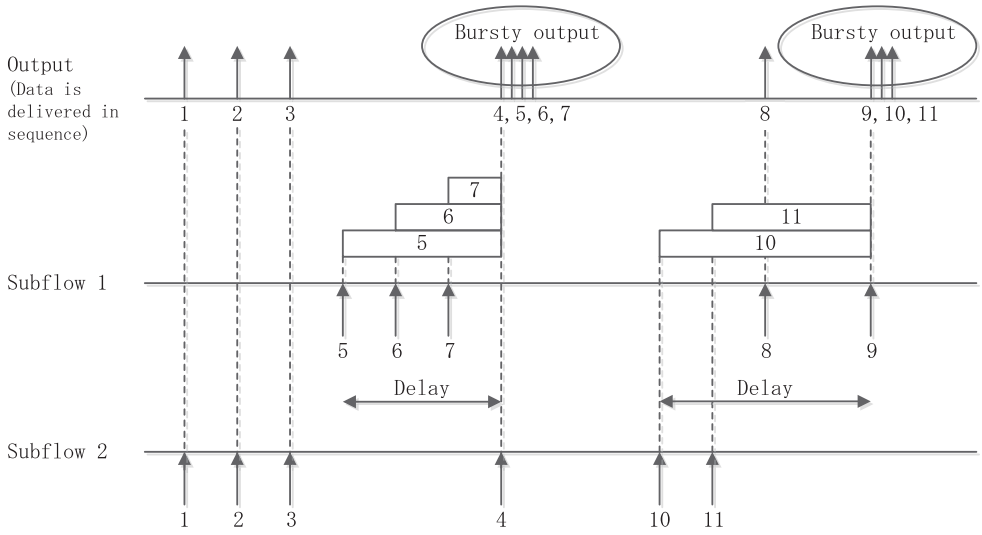

Fig. 5 An example of resequencing sequence

2.4 MPTCP の研究課題と本研究の目的

MPTCP で複数経路通信による性能を引き出すためには, 所要受信バッファ量, 遅延時間, バースト性などを把握することは重要と考えられる. MPTCP の性能に関して従来から無線ネットワークに適用した場合の性能評価がなされているが[10][11], 受信側の順序制御に着目した検討や評価がなされていない. また, MPTCP をネットワークシミュレータ ns-3 に実装した論文が発表されているが[12], ここでは主に MPTCP の輻輳制御に着目して研究が行われている. 本研究は, MPTCP の受信側における順序制御に着目して所要受信バッファ量, 遅延時間, バースト性などをシミュレーションにより評価することを目的としている.

3. ns-2 によるシミュレーション

3.1 ns-2 によるシミュレーションモデル

本研究では, ネットワークシミュレータ ns-2[14] を使用して MPTCP の性能評価を行う. Fig. 6 はシミュレーションモデルを示している.

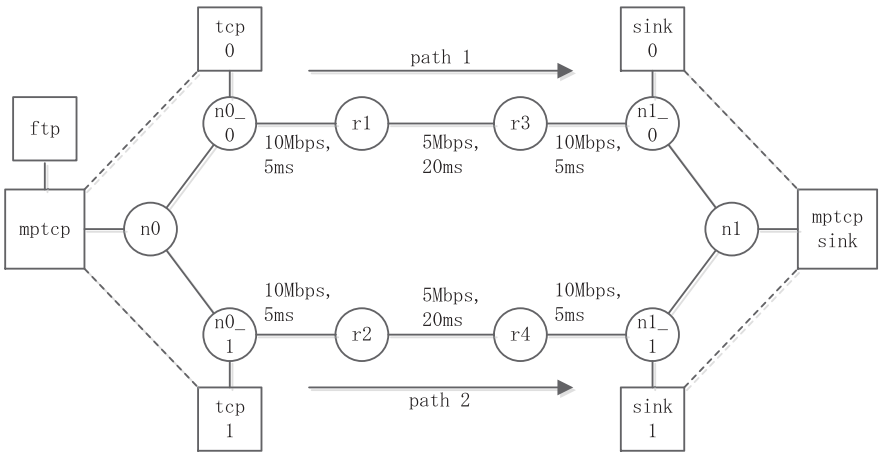

Fig. 6 Network configuration of MPTCP simulation by ns-2

このモデルで MPTCP は 2 本の経路（path1，path2）を使用して送信側から受信側へ，データを転送する．本研究では path1 の遅延時間と回線速度を固定とし，path2 の遅延時間，回線速度，パケット損失率をパラメータとして変化させ評価を行う．ns-2 では MPTCP が標準的にはサポートされておらず，インターネットで公開されている MPTCP のプログラム[15]を ns-2 に追加して評価を行った．シミュレーションパラメータ一覧を Table 1 に示す．

Table 1 Simulation parameters

Simulation time	30s	
TCP	Type	Sack
	Window size	53.6kB
Path1	Bandwidth	5Mbit/s
	Delay	20ms + 10ms
	Packet loss rate	0
Path2	Bandwidth	2, 5, 10Mbit/s
	Delay	20, 50, 100ms + 10ms
	Packet loss rate	0, 0.0001, 0.0002, 0.0005, 0.001, 0.002, 0.005, 0.01

ns-2 のプログラムは C++によるオブジェクト指向で開発されており，全体が階層的なクラス構成を取っている．MPTCP のプログラムのクラス構成を Fig. 7 に示す．MPTCP は基本的に既存の TCP モジュールを利用して実現するが，ns-2 では多種類の TCP プログラムが存在するものの，ほとんどが片方向のデータ転送しか実現していない．MPTCP をシミュレートするためには，MPTCP 独自のプロトコル情報を転送するために，両方向のデータ転送機能が必要で，ns-2 に備わった TCP プログラムでは SackFullAgent のみ両方向が可能である．これから，使用したシミュレーションプログラムでは SackFullAgent から派生クラス MpFullTcpAgent を定義し，このクラスの複数オブジェクトを MptcpAgent が束ねる形で実現している．従って，シミュレーションでは Sack を用いる TCP が下位に用いられる．図に示すようにシミュレーションの実装上，SackFullAgent は最大 100 本の MpFullTcpAgent による TCP コネクションを利用できる．

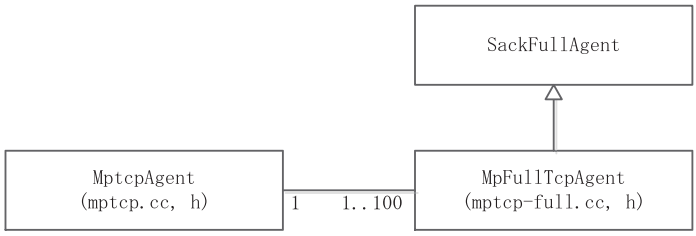

Fig. 7 Class diagram of MPTCP program in ns-2

3.2 受信側での順序制御

使用した MPTCP のプログラムでは，受信側での順序制御が実装されていない問題がある．このため，Fig. 8 に示すようにシミュレーションプログラムの外部で，ns-2 が出力するトレースファイルを Python のプログラムで分析して順序制御の影響を評価した．評価項目としては，順序制御後のスループット，遅延時間，バースト長がある．Python によるプログラムでは，宛先ノードのセグメント受信イベントを順次取り出し，イベントに含まれる MPTCP のシーケンス番号を検査し，次に受信を期待する番号ではない場合，受信したセグメントをバッファに保留する．次に受信を期待する番号の場合は，受信セグメントを出力するとともに，バッファに保留されているセグメントの中で順序が正しく揃っているセグメントを出力し，さらに次に受信を期待するシーケンス番号を更新する．

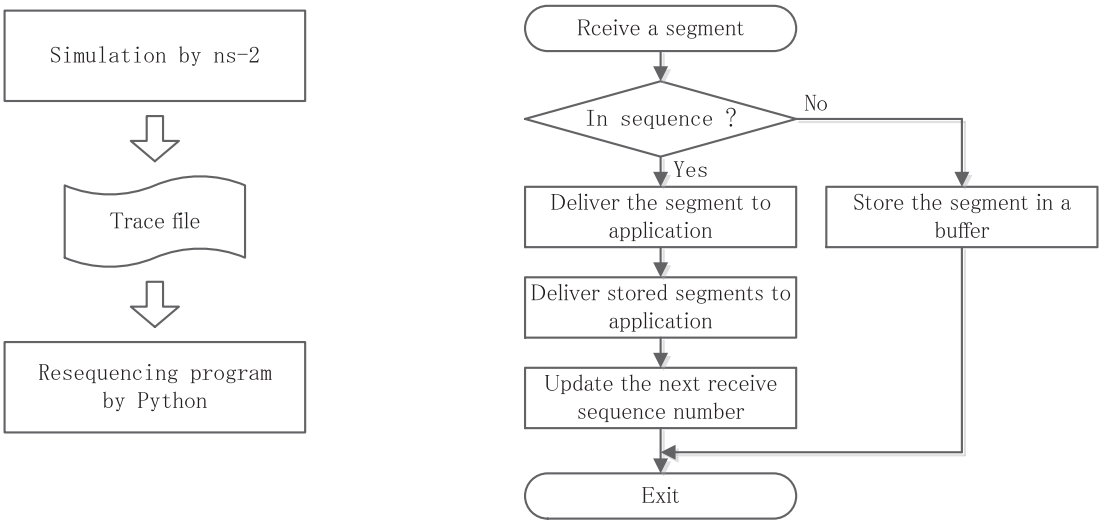

Fig. 8 Resequencing procedure for the trace file generated by ns-2 simulation

3.3 受信バッファ制限

MPTCP では受信側が受信可能なバッファ量を送信側に通知し，送信側は新規セグメントの送信をこれに合わせて制限することにより，受信バッファ溢れが生じないような制御が行われる．残念ながら，使用した MPTCP のプログラムでは，受信側がバッファ量を通知し，送信側がこれに合わせて新規送信を制御する機能が実現されていない．このため，実質的に受信バッファ量に制限のないシミュレーションとなる．

このため，シミュレーションプログラムで新規送信を行う関数の一部を下記のように修正した．具体的には，mptcp.cc のファイルにおいて，関数 MptcpAgent::send_control () 内で新規のデータ送信処理を行う部分にコードを追加した．これは送信側が送達確認していないデータ量が，予めパラメータとして決めた受信バッファ量 rbuf を越えた場合は，新規の送信を行わないようにするものである．mcurseq_は送信した最大のシーケンス番号，mptcp_ack は送達確認された最大のシーケンス番号を示し，両者の差は送信したが未だ送達確認が取れていないデータ量を示す．このデータ量は，正確には受信側で順序制御のために保留されているデータ量とは異なり，これよりも大きめの値を取るが近似的に受信バッファ制限をシミュレートしていると考えられる．rbuf の値はシミュレーションのパラメータとし，シミュレーション中は変更しない．

```
while (sendbytes >= mss) {

   if ((mcurseq_ - mptcp_ack) > rbuf) break;    // modification for receive buffer limit

   subflows_[i].tcp_->mptcp_add_mapping (mcurseq_, mss);
   subflows_[i].tcp_->sendmsg (mss);
   mcurseq_ += mss;
   sendbytes -= mss;
}
```

4. シミュレーション結果

4.1 パケット損失が無い場合

Fig. 9 は経路 2 の遅延時間（ノード r2 と r4 間の回線の遅延時間），Fig. 10 は経路 2 の帯域（ノード r2 と r4 間の回線速度）を変化させた場合のスループットを示している．ここで，スループットは経路 1 と経路 2 の合計速度で除した規格化した値を使用し，順序制御前の下位層の TCP のスループットと順序制御後のスループットの両方を示している．どちらの場合もパケット損失は発生しないものとしている．遅延時間や回線速度が二つの経路で異なっても，経路の合計帯域に近い性能の得られることが分かる．遅延時間差や回線速度差が大きくなるほどスループットが低下する傾向が見られるが，下位の TCP におけるウィンドウが不足していることによる影響と考え

られ，TCP のウィンドウを増加すれば改善できると考えられる．順序制御前と順序制御後でスループットの値は変わらず，順序制御がスループットに与える影響はないことが確認できる．

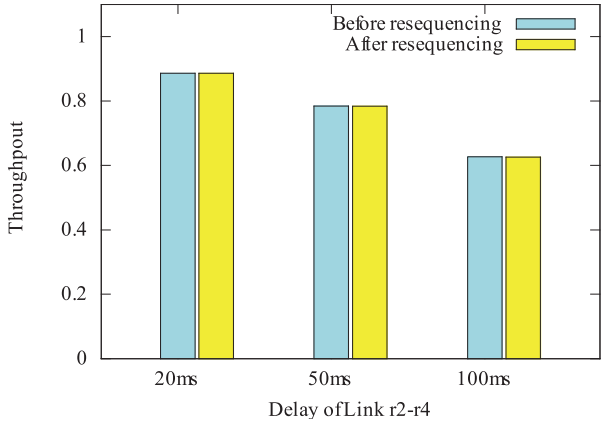

Fig. 9 Throughput vs. path delay

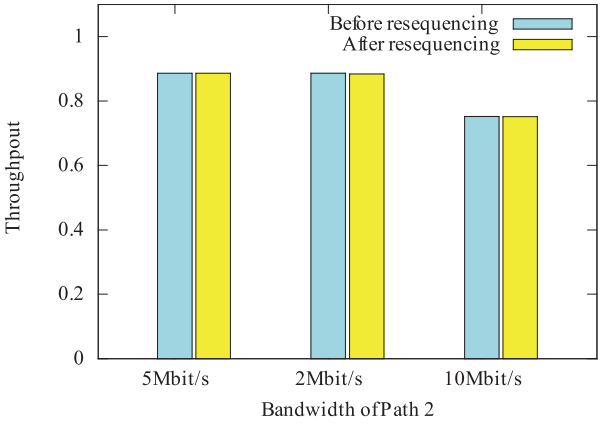

Fig. 10 Throughput vs. path bandwidth

Fig. 11 と Fig. 12 はそれぞれ Fig. 9 と Fig. 10 の場合の出力最大バースト長を示している．二つの経路における，遅延時間や帯域の差が大きいほど大きなバーストが発生することが分かる．

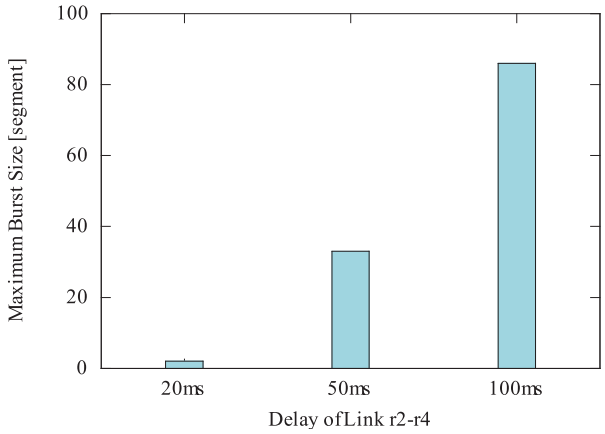

Fig. 11 Max. burst size vs. path delay

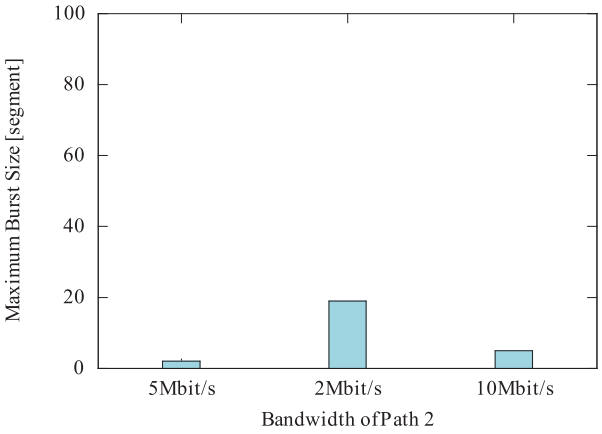

Fig. 12 Max. burst size vs. path bandwidth

Fig. 13 は経路 2 の帯域が 5Mbit/s，遅延時間が 100ms の場合におけるバースト的なデータ出力（セグメント単位）の時間変化を示している．

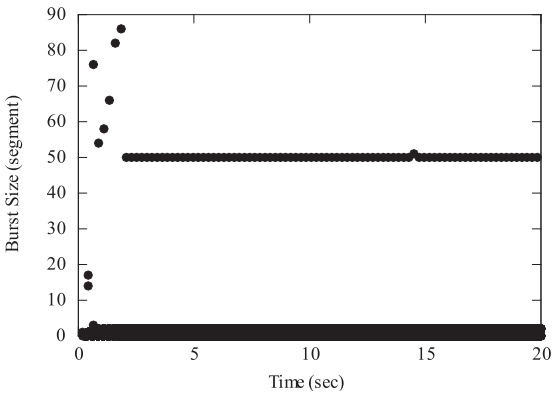

Fig. 13 Generation of bursts (0 to 30sec)

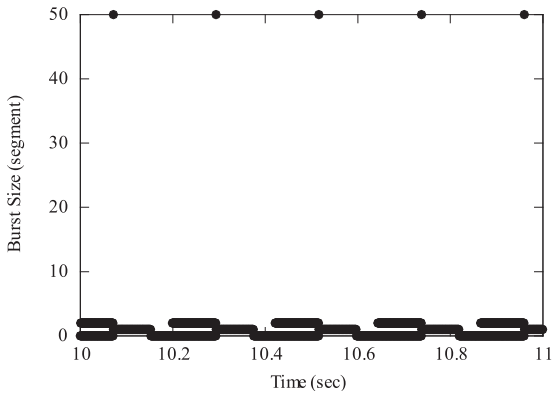

Fig. 14 Generation of bursts (10 to 11sec)

開始直後には 90 セグメント近いバーストが発生しているが，3 秒を経過した後は定常的に 50 セグメント程度のバーストが発生していることが分かる．Fig. 14 は Fig. 13 において，10 秒から 11 秒の時間間隔のバースト出力を拡大表示したものである．大きさが 50 セグメントのバーストが約 0.2 秒ごとに発生し，それ以外では出力バーストの大きさは高々2 セグメント程度であることが分かる．パケット損失が無い状況ではバーストの発生は周期的な振る舞いが見られる．

4.2 受信バッファ制限の影響

Fig. 15 はパケット損失率 P_L を 0 として，バッファの大きさを変化させた場合のスループット，Fig. 16 は順序制御による遅延時間を示している．ここで，経路 2 の帯域と遅延時間は経路 1 と同じとしている．パケット損失は乱数によりランダムに発生するものとし，乱数の初期値を変えたシミュレーションを 12 回行った平均値と標準偏差を求め表示している．パケット損失率 P_L が 0 の場合，乱数の影響を受けないため標準偏差は極めて小さい．バッファが 65 k バイトの場合はバッファ不足によりスループットが小さいが，これよりバッファが大きくなるとほぼ一定のスループットが得られる．バッファが 65 k バイトの場合は，順序制御による遅延時間も大きくなっている．

Fig. 17 パケット損失率 P_L が 0.0001 の場合にバッファの大きさを変化させた場合のスループット，Fig. 18 は遅延時間を示している．どちらの図でも平均値と測定値のバラツキを評価するための標準偏差を示している．バッファの大きさが 130K 以上でスループットが得られている．バッファの大きさが 1000k バイトを超えると遅延時間が大きくなることが分かる．

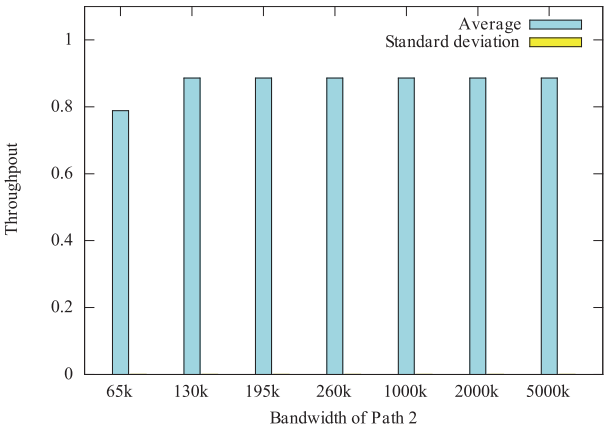

Fig. 15 Throughput vs. buffer size (P_L = 0)

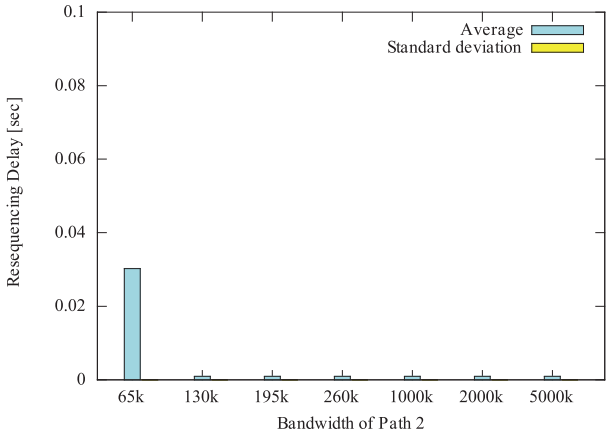

Fig. 16 Delay vs. buffer size (P_L = 0)

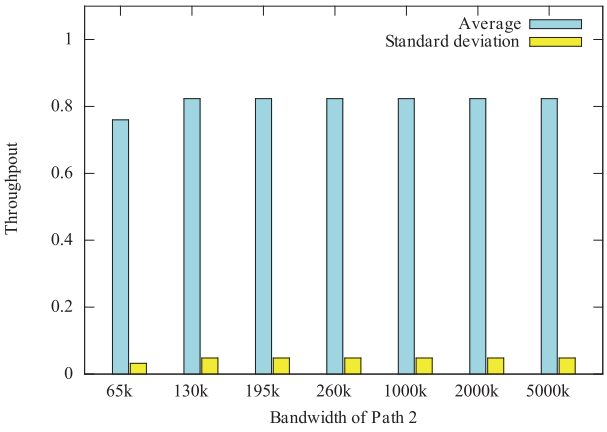

Fig. 17 Throughput vs. buffer size (P_L = 0.0001)

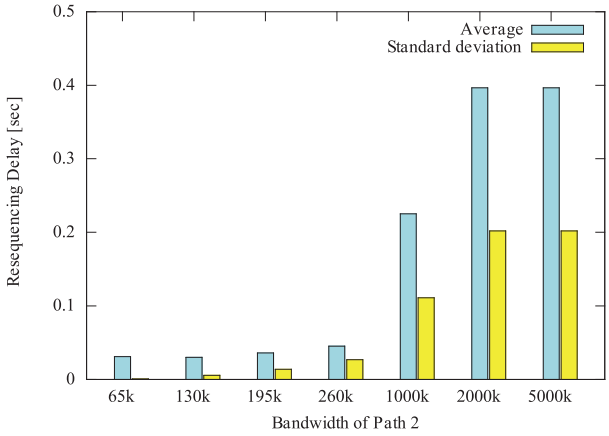

Fig. 18 Delay vs. buffer size (P_L = 0.0001)

4.3 パケット損失率の影響

Fig. 19 はパケット損失率を変化させた場合のスループットで，受信バッファ制限がない場合（No buffer limit）と受信バッファを 260k バイトに制限した場合の両方を示している．受信バッファ制限がない場合，0.0005 より大きなパケット損失率ではスループットの改善が見られるが，0.0002 より小さなパケット損失率では受信バッファ制限した場合と同程度のスループットとなっている．Fig. 20 は受信バッファ制限がない場合とある場合での最大バースト長，Fig. 21 は平均遅延時間を示している．受信バッファの制限がない場合，順序制御による最大バースト長および平均遅延時間が受信バッファ制限する場合に比べ著しく大きいことが分かる．これから，受信バッファ量を適切に制限することが必要と言える．

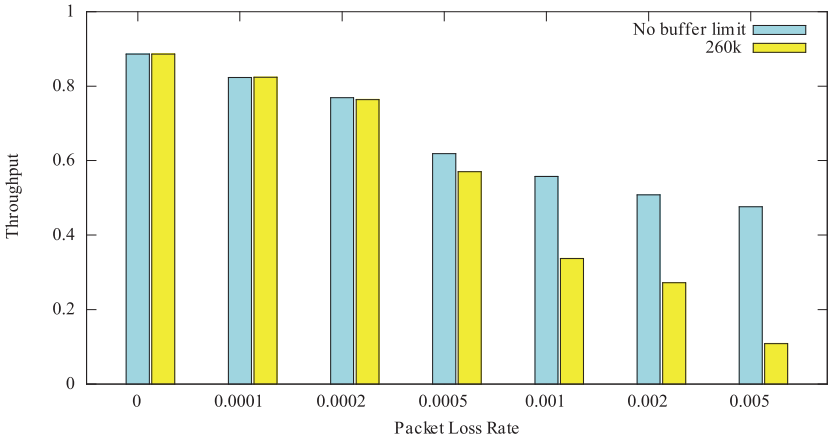

Fig. 19 Throughput vs. packet loss rate over path 2

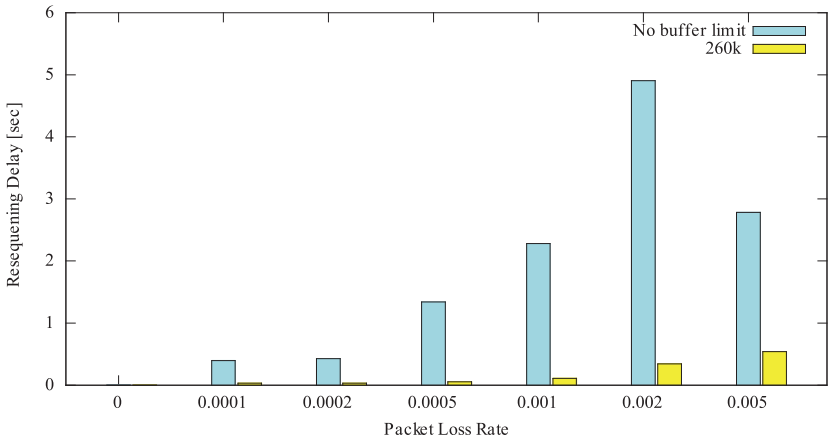

Fig. 20 Burst size vs. packet loss rate over path 2

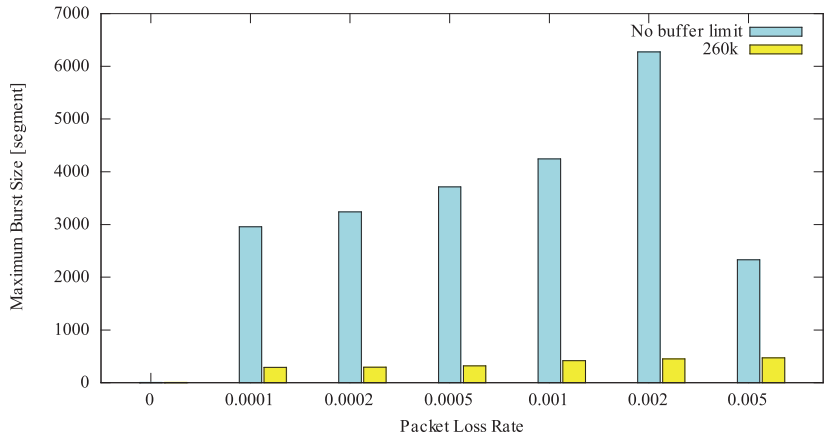

Fig. 21 Resequencing delay vs. packet loss rate over path 2

5. 結　　言

本論文では MPTCP の受信側における順序制御に注目し， MPTCP の性能について ns-2 によりシミュレーション評価を行った．これから，MPTCP が複数経路の帯域を利用できること，順序制御により受信バッファが必要であること，順序制御により受信バッファ量を大きくすればパケット損失率が大きい場合のスループットが改善されること，大量の受信バッファを用いると順序制御による遅延時間やバースト的な出力が大きくなるため，数 100k バイト程度の受信バッファ量に制限する必要のあることが分かった. 本研究で使用した ns-2 ではプロトコルが簡略化された部分があり，より厳密にプロトコルが実装されているネットワークシミュレータ ns-3[16][17]によるより詳細なシミュレーション評価や Linux を用いた実験評価が今後の課題となる．さらに，ns-3 上で実際の Linux のプロトコルスタックを実行できる DCE (DIRECT CODE EXECUTION)[18]を用いた評価も予定している．

謝　　辞

本研究は，福井工業大学特別研究費の助成を受けたものである．ここに記して謝意を表す.

文　　献

(1) 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”，オーム社, 2015 年 2 月.
(2) Apple, “iOS : iOS 7 における Mutlipath TCP のサポート” , https://support.apple.com/ja-jp/HT201373 (参照日 2016 年 1 月 25 日).
(3) icteam, “MultiPath TCP - Linux Kernel implementation”, http://www.multipath-tcp.org/ (参照日 2016 年 1 月 25 日).
(4) R. Stewart, “Stream Control Transmission Protocol”, RFC4960, Sept. 2007.
(5) ars technica, “Multipath TCP lets Siri seamlessly switch between Wi-Fi and 3G/LTE”, http://arstechnica.com/apple/2013/09/multipath-tcp-lets-siri-seamlessly-switch-between-wi-fi-and-3glte/(参照日 2016 年 1 月 25 日).
(6) J. Border, M. Kojo, J. Griner, G. Montenegro and Z. Shelby, “Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations” , RFC3135, June 2001.
(7) A. Ford, C. Raiciu, M. Handley and O. Bonaventure, “TCP Extensions for Multipath Operation with Multiple Addresses”, RFC6824, Jan. 2013.
(8) A. Ford, C. Raiciu, M. Handley, S. Barre and J. Iyengar, “Architectural Guidelines for Multipath TCP Development”, RFC6182, March 2011.
(9) Y. Nishida, “Multipath TCP の紹介と最近の動向”, https://www.isoc.jp/materials/20131220/20131220_mptcp.pdf#search='MPTCP' (参照日 2016 年 1 月 25 日).
(10) Y. Chen, Y. Lim, R. Gibbens, E. Nahum, R. Khalili and D. Towsley, “A measurement-based study of MultiPath TCP performance over wireless networks”, IMC’13, pp. 455-468, 2013.
(11) S. Deng, R. Netravali, A. Sivaraman and H. Balakrishnan, “WiFi, LTE, or Both? Measuring Multi-Homed Wireless Internet Performance”, IMC’14, pp. 181-194, Nov. 2014.
(12) M. Kheirkhah, I. Wakeman and G. Parisis, “Multipath-TCP in ns-3”, http://users.sussex.ac.uk/~mk382/mptcp-wns3-2014.pdf#search='MultipathTCP+in+ns3' (参照日 2016 年 1 月 25 日).
(13) 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” , 森北出版, 2011 年 9 月.
(14) VINT project, “The Network Simulator - ns-2”, http://www.isi.edu/nsnam/ns/ (参照日 2016 年 2 月 20 日).
(15) Y. Nishida, “MPTCP implementation on ns-2”, http://www.jp.nishida.org/mptcp/ (参照日 2016 年 1 月 20 日).
(16) “ns-3”, https://www.nsnam.org/ (参照日 2016 年 2 月 20 日).
(17) 銭 飛, “ns3 によるネットワークシミュレーション”，森北出版, 2014 年 1 月.
(18) “DIRECT CODE EXECUTION”, https://www.nsnam.org/overview/projects/direct-code-execution/ (参照日 2016 年 2 月 20 日).

（平成 28 年 3 月 31 日受理）