遅延時間制御手法の動的調整による TCP 公平性の向上

花井 雅人[1]　山口 実靖[1]　小林 亜樹[1]

概要: 近年のルータのバッファの大容量化が原因で，恒常的にネットワーク遅延時間が大きくなる Bufferbloat 問題が指摘されている．そして，その解決策として遅延時間の観測による待ち行列制御手法が提案されており，今後は本手法が普及していくと期待できる．また，近年の多数の新しい TCP アルゴリズムの提案により TCP 公平性という問題が生じており，我々は過去に上記の遅延時間制御手法の改変による TCP アルゴリズム間性能公平性向上手法を提案している．しかし，この公平性改善手法はネットワーク帯域を大きく消費する接続が既知であるとことを仮定しており，これが既知でない環境には適用することができない．本稿では，通信機器にてトラフィックを観察し，ネットワーク帯域消費の大きい接続を推定し，その接続のパケットを優先的に破棄することにより TCP 公平性を改善する手法を提案する．そして，提案手法を実機に実装し，性能評価によりその有効性を示す．

TCP Fairness Improvement with Dynamically Adjusting CoDel

Masato HANAI[1]　Saneyasu YAMAGUCHI[1]　Aki KOBAYASHI[1]

1. はじめに

TCP は現在のインターネットにおいて標準的に用いられているトランスポート層プロトコルである．TCP にはネットワークの輻輳制御機能が実装されており，本機能の動作が通信性能に大きな影響を与える．従来の TCP 実装では輻輳制御アルゴリズムとして TCP Reno[1]が使用されてきたが，古典的な TCP アルゴリズムである TCP Reno では高遅延・広帯域のネットワーク環境においてネットワーク帯域を十分に使い切ることができない問題が指摘されており[2][3][4]，この問題の解決策として BIC TCP[5]，CUBIC TCP[6]，Compound TCP[7]などの新しい高速 TCP が多数提案された．

これらの複数の新しい TCP アルゴリズムの提案により，TCP アルゴリズム間の性能公平性(TCP 公平性)という新しい課題が生じた[8][9]．Linux OS には CUBIC TCP が搭載されており，Windows OS には Compound TCP が搭載されているが，両者が同一ネットワークにおいて同時に通信を行った場合，通信性能に著しい不公平な問題があることがシミュレーションによる検証[8][10]や実機を用いた検証[9][11]にて確認されている．

TCP 公平性の改善手法として，RED[12]の適用や，RED を改善した Active Packet Dropping[13]手法があり，これにより公平性が改善することが確認されている．ただし，RED は設定するパラメータが多く設定が困難であることなどから必ずしも普及していない．また，設定がより容易である遅延時間制御手法(CoDel)[14]が提案され，今後普及していくとことが考えられるよって，TCP 公平性改善の手法などは遅延時間制御手法に基づくものを考察していくことが重要であると考えられる．我々は過去に，CoDel を改変し TCP アルゴリズム間性能公平性向上手法を提案している[15]．しかし，この公平性改善手法はネットワーク帯域を大きく消費する接続が通信機器にとって既知であるとことを仮定しており，これが既知でない環境には適用することができないという課題を抱えている．本稿では，通信機器にてトラフィックを観察し，ネットワーク帯域を大きく消費する接続を推定し，その接続のパケットを優先的に破棄することにより TCP 公平性を改善する手法を提案する．そして，提案手法を実機に実装し，性能評価によりその有効性を示す．

2. 関連研究

2.1. TCP 輻輳制御アルゴリズム

過剰なパケットを送出しネットワークの輻輳を招くことを避けるために，TCP 実装には輻輳制御機能が搭載されている．TCP によるパケット送出量はこの輻輳制御アルゴリズムにより制限されており，TCP を用いる通信の性能はこのアルゴリズムに大きな影響を受ける．OS により様々な輻輳制御アルゴリズムが実装されおり，それぞれの OS で輻輳制御の仕方が異なる．

TCP 輻輳制御手法は主に，ロスベース手法，遅延ベース手法，両者を組み合わせたハイブリッド型の手法の 3 つに分類できる．

ロスベース手法はパケットが破棄されたのを検出し，これに基づいて輻輳ウィンドウを制御する手法である．通常時は確認応答を受信するたびに輻輳ウィンドウを増加させ，パケット破棄が検出された時には輻

1 工学院大学大学院 工学研究科 電気・電子工学専攻
Electrical Engineering and Electronics, Kogakuin University Graduate School

輳ウィンドウを減少させる．従来の TCP である TCP Tahoe，TCP Reno や，近年の Linux OS で使用されている BIC TCP[5]，CUBIC TCP[6]がロスベース手法である．

遅延ベース手法は，RTT の増減に基づいて輻輳ウィンドウを制御する手法である．ロスベース手法の様に輻輳が発生してから速度を減少させるのではなく，RTT の増加からネットワークの混雑状況を推定し，輻輳が発生する前に速度を減少させるため安定した通信速度を期待することができる．欠点としてロスベース手法と同一ネットワークにおいて同時に通信を行ったときに得られる性能が低くなってしまうということが指摘されている[8][15]．代表的な遅延ベース手法に TCP Vegas[16]がある．

ハイブリッド型手法はロスベースと遅延ベースを組み合わせた手法である．代表的なハイブリッド型手法の TCP に Windows 系 OS に搭載されている Compound TCP[6]がある．

本研究では，Linux OS の標準 TCP である CUBIC TCP と Windows 系 OS に搭載されている Compound TCP に焦点を当て考察を行う．

2.2. CUBIC TCP

CUBIC TCP[6]は，BIC TCP[5]のスケーラビリティを維持しながら，既存の TCP アルゴリズムとの公平性である TCP-Fairness や，RTT の異なる通信間での公平性である RTT-Fairness，制御手法の複雑さを改善した高速 TCP アルゴリズムである．CUBIC TCP は Linux2.6.19 以降で標準の TCP として搭載されている．

CUBIC TCP では，BIC TCP のバイナリーサーチを用いて利用可能帯域を検索するアルゴリズムを式(1)，(2)のような 3 次関数を用いた制御によって実現している．その輻輳ウィンドウの推移は図 1 のようになる．

$$cwnd = C(t-K)^3 + W_{max} \qquad (1)$$

$$K = \sqrt[3]{\frac{W_{max}\beta}{C}} \qquad (2)$$

ここで，$cwnd$ は輻輳ウィンドウサイズ，t はパケットロス検出時からの経過時間，W_{max} はパケットロス検出時の輻輳ウィンドウサイズ，C は増加幅を決めるパラメータ，β はパケットロス検出時のウィンドウサイズ減少幅を表している．通常，C には 0.4 が，β には 0.2 が用いられている．

CUBIC TCP では，上記のようにパケットロス検出時からの経過時間を用いて輻輳ウィンドウの値を定めている．これは，RTT の影響を強く受ける ACK の受信を輻輳ウィンドウサイズの増加処理から排していることを意味し，これにより RTT-Fairness が向上することが期待できる．加えて，BIC TCP の低遅延環境で輻輳ウィンドウサイズを急速に成長させすぎる問題もこれにより解決している．

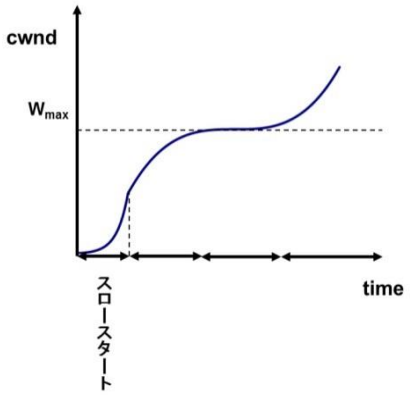

図 1. CUBIC TCP の輻輳ウィンドウの推移

また，TCP Reno を用いた場合に得られる輻輳ウィンドウサイズを式(3)により計算し，現在のウィンドウサイズが計算値よりも小さい場合はその計算値を輻輳ウィンドウサイズとして採用している．これにより，TCP-Fairness の向上を目指している．

$$cwnd = W_{max}(1-\beta) + 3\frac{\beta}{2-\beta}\frac{t}{RTT} \qquad (3)$$

また，ボルトネックを共有するフローが帯域を公平に分け合うまでの収束時間を短縮するため，パケットロスを検出した時の輻輳ウィンドウサイズ値が前回ロスを検出した時の値を下回っている場合，新たな W_{max} は次式のように設定される．

$$W_{max} = cwnd \times (2-\beta)/2 \qquad (4)$$

以上のようなしくみにより，CUBIC TCP は高いスケーラビリティ，RTT-Fairness，TCP-Fairness を目指している．

2.3. Compound TCP

Compound TCP はロスベースの輻輳制御で動作するロスウィンドウと遅延ベースの輻輳制御で動作する遅延ウィンドウを両方使ったハイブリッド型の TCP アルゴリズムである．

ロスウィンドウはスロースタートフェーズと輻輳回避フェーズという 2 つのフェーズで構成されている．この 2 つのフェーズでウィンドウの増加量が異なり，それぞれ次式で与えられる．

$$cwnd(t+1) \leftarrow \begin{cases} cwnd(t)+1 & (\text{スロースタートフェーズ時}) \\ cwnd(t)+\dfrac{1}{swnd(t)} & (\text{輻輳回避フェーズ時}) \end{cases} \qquad (5)$$

ただし $swnd$ は現在のロスウィンドウと遅延ウィンドウの和であり，t は ACK を受信する度に増加する値であり実時間に依存しない．スロースタートフェーズではロスウィンドウを指数関数的に増加させ，輻輳回避フェーズではロスウィンドウを線形的に増加させる．

パケットロスを検出した場合，ロスウィンドウが減

少する．この減少量はパケットロスの検出方法によって異なり，それぞれ次式で与えられる．

$$cwnd\ (t+1) \leftarrow \begin{cases} \dfrac{cwnd\ (t)}{2} & (\text{重複 } ACK \text{ による検出}) \\ 1 & (\text{タイムアウトによる検出}) \end{cases} \quad (6)$$

重複 ACK を受信した場合のパケットロスは，ネットワークにおいて軽度な輻輳が発生したと判断してロスウィンドウを現在の半分の値まで減少させる．一方，タイムアウトによるパケット破棄を検出した場合はネットワークにおいて重度な輻輳が発生したと判断してロスウィンドウを 1 に減少させる．また，遅延ウィンドウもスロースタートフェーズと輻輳回避フェーズにより動作が異なる．遅延ウィンドウはスロースタートフェーズにおいては動作せず，輻輳回避フェーズにおいてのみ動作する．

遅延ウィンドウサイズは，ネットワーク中に滞留しているパケット数 $Diff$ に基づいて次式で与えられる．

$$dwnd(t+1) = \begin{cases} dwnd(t) + (\alpha \cdot swnd(t)^k - 1) & (Diff < \gamma) \\ (dwnd(t) - \zeta \cdot Diff) & (Diff \geq \gamma) \end{cases} \quad (7)$$

$$\begin{aligned} Expected &= \frac{swnd\ (t)}{baseRTT} \\ Actual &= \frac{swnd\ (t)}{RTT} \\ Diff &= (Expected - Actual) \cdot baseRTT \end{aligned} \quad (8)$$

ここで，$baseRTT$ は実際に観測された往復遅延時間の最小値，RTT は現在の往復遅延時間である．

推測値 $Diff$ が閾値 γ よりも小さい場合には，ネットワークに未使用の帯域があると判断し，遅延ウィンドウを増加させる．推測値 $Diff$ よりも閾値 γ が大きい場合にはネットワークに輻輳が発生していると判断し，遅延ウィンドウを減少させる．

Compound TCP ではロスウィンドウおよび遅延ウィンドウを用いて送出ウィンドウを次式に基づいて決定する．

$$swnd(t+1) = cwnd(t+1) + dwnd(t+1) \quad (9)$$

2.4. RED (Random Early Ditection)

RED は平均待ち行列長に応じた確率でパケットを破棄する手法である．Taildrop では待ち行列がバッファサイズに達するとすべてのコネクションのパケットが破棄される．一方 RED では平均待ち行列長が閾値 min_{th} に達するとパケットの破棄が開始され，平均待ち行列長が max_{th} に達すると破棄率が 1 となり，到着したパケットすべてが破棄される．RED では安定した輻輳制御が行うことができ，公平性の改善も実現できると期待されている．

図 2 に RED における平均待ち行列長に対する破棄率を示す．minth, maxth, maxq は制御用パラメータである．平均待ち行列長は Wq を用いて次式で与えられる．

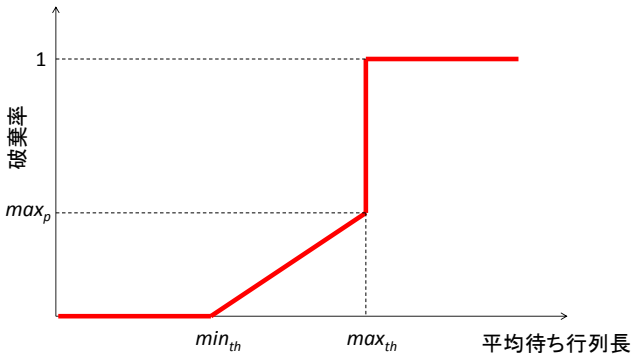

図 2　RED

$$\bar{q} = (1 - W_q)\bar{q} + W_q q \quad (10)$$

2.5. Active Packet Dropping

文献[13]にて，スイッチやルータなどのネットワーク機器におけるパケット破棄を制御することにより TCP 公平性を改善させる手法である Active Packet Dropping が提案されている．当該手法は RED を基にしており，帯域消費の大きいコネクションの RED におけるパケット破棄確率を増加させ，公平性の改善を図っている．

両文献では，静的手法と動的手法が提案されており，静的手法では帯域消費が大きいコネクションの情報を既知である前提のもと，そのコネクションのパケット破棄確率を増加させる．動的手法では，定期的にパケットを監視し，サンプリングされたパケットから帯域消費の大きいコネクションを推定し，そのコネクションのパケット破棄確率を増加させる．

両手法の実装と実機，実 TCP 実装を用いた性能評価がなされ，公平性の改善が確認されている．

2.6. Bufferbloat

近年のルータなどの通信機器には大容量のバッファが搭載されている．現状のインターネットのルータのバッファは常にパケットで満たされており，各パケットは長い待ち行列を待つ必要があり，結果として現状のインターネット通信の通信遅延は常に大きい状態にあるとの指摘[17][18]があり，この問題は Bufferbloat と呼ばれている．Bufferbloat の解決策の一つとして，次節で述べる遅延時間制御手法(CoDel)[14]がある．

2.7. CoDel (Controlling queue Delay)

CoDel[14]はルータ待ち行列のスケジューリングアルゴリズムである．CoDel は，あるパケットが待ち行列に入ってから待ち行列を出るまでの時間が target 時間以上になると，パケットを破棄する．target はチューニングパラメータであり，初期値は 5ms である．

RED と異なり CoDel は設定(チューニングパラメー

タ)が少なく，パケット破棄するか否かは待ち行列の滞在時間のみで決定される．RED の普及がかならずしも進まない原因の一つとしてパラメータの多さと，それらの設定の難しさがあげられ，CoDel ではより簡単な設定で高い性能を実現でき，今後は普及が進んでいくと期待することができる．

RED と同様に，CoDel にてルータ上のパケット待ち行列を制御することにより TCP 公平性の改善が実現できると考えられ，公平性の改善など関しては今後は本手法を基にした考察を行うことが重要になっていくと考えられる．

2.8. 静的優先制御に公平性改善手法

前節で述べたように，CoDel の適用により TCP 公平性の改善ができると期待できる．我々は文献[15]において CoDel 適用環境と非適用環境における TCP 公平性の評価を行い，一部の環境にて CoDel の適用により TCP 公平性の改善が可能であること，一部の環境では CoDel を適用しても公平性が低いことを示した．そしてネットワーク帯域を大きく消費する接続が通信機器にとって既知であることを前提とし，この前提の元で TCP アルゴリズム間性能公平性を向上する手法を提案し，性能評価により公平性の改善を示した．

しかし，一般にネットワーク帯域を大きく消費する接続は通信機器にとって既知ではなく，この前提なく適用可能な手法の考察が重要な課題となっている．

3. CUBIC TCP と Compound TCP 公平性の評価

本章では，実ネットワーク上で TCP 公平性(異なる TCP 輻輳制御アルゴリズム間の性能公平性)についての評価を行う．

図 3 のネットワークを構築し，CUBIC TCP コネクションと Compound TCP コネクションが混在する環境における通信速度を iperf [19]を用いて測定した．Linux 1，Linux 2，Windows の 3 台の計算機の仕様は表 1 の通り，CoDel を搭載した CoDel 機およびネットワークの遅延をエミュレートする Delay 機の仕様は表 2 の通りである．Delay 機は人工的にネットワーク遅延時間を発生させる装置であり，Linux Netem を用いて構築した．

ネットワーク機器は全て 1Gigabit Ethernet に対応している．

通信は，Linux1-Linux2 間と，Windows-Linux2 間で同時に行い，Linux1 と Windows を送信端末，Linux2 を受信端末とした．Linux1-Linux2 間の通信と，Windows-Linux2 間の通信は CoDel 機から Delay 機，Linux2 までのネットワークを共有している．Delay 機における人工的な遅延時間は 2ms から 64ms に変動させて測定を行った．それぞれの受信ウィンドウサイズは 32MB とした．

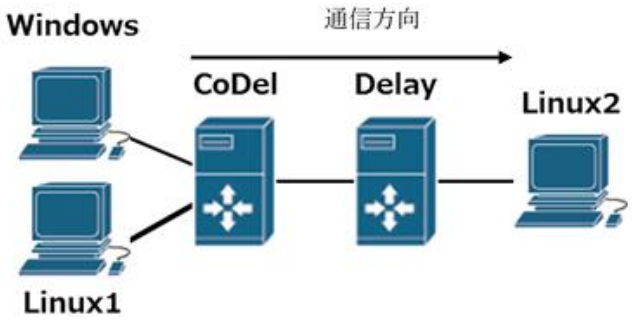

図 3　ネットワーク構成

表 1　計算機仕様

CPU	AMD Turion, 2.20[GHz]
メモリ	4[GB]
OS	(Linux1) Linux 4.2.3 (Linux2) Linux 3.3.4 (Windows) Windows7 Enterprise

表 2　計算機仕様

CPU	Intel CelronG540, 2.4[GHz]
メモリ	2[GB]
OS	(Codel, Delay) Linux 3.17.4

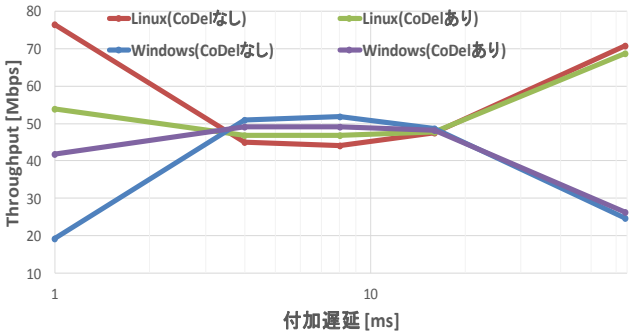

図 4　評価結果

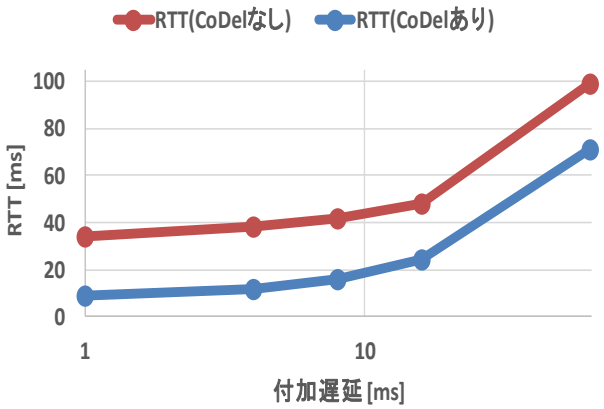

図 5　評価結果(送信者-受信者間の RTT)

測定結果を図 4，図 5 に示す．横軸の値は Delay 機により付加したネットワーク内における送信者-受信者間の往復遅延時間である．縦軸は各通信が得た通信速度で 10 コネクションの平均値である．図より，CUBIC TCP の通信速度と Compound TCP の通信速度には大きな差があり，TCP 公平性が非常に低いことが確認できる．特に付加遅延時間が 64[ms]において公平性が低くなっている．

また，図 5 より Codel の適用により大きな通信速度の劣化なく通信遅延時間の減少が実現できていることが確認できる．

4. 提案手法

4.1. 公平性向上遅延制御手法

本章にて，CoDel に基づく TCP 公平性の改善手法を提案する．CoDel ではパケットの待ち時間が指定値 `target` を超えると必ずパケットの破棄を行うが，本稿では通信帯域の消費が最も大きいと予想されるコネクションのパケットは必ず破棄し，それ以外のパケットは確率 p で破棄する手法を提案する．p はチューニングパラメータである．

通信帯域の消費が最も大きいコネクションの予想は，以下の様に行う．通信機器を通過するパケットのうち *stat_int* 個に 1 個をログに記録する．そして 100 個の記録が行われるたびにログを調査し，最も多く登場したコネクションを，通信帯域の消費が最も大きいコネクションと予想する．

4.2. 実装

Linux に実装されている CoDel 実装を修正することにより提案手法の実装を行った．本稿では，本研究の初期研究として以下の様な簡易な実装を行い，性能評価を行った．コネクションの管理は送受信端末の MAC アドレスにより行った．確率 p での破棄は乱数により無記憶的に決定することはせず，ループするカウンタを用い，$1/p$ 回に 1 回破棄を実行する様にした．p=50%の例では破棄を行うと行わないを交互に選択する．

5. 性能評価

提案手法の有効性を検証するために，提案手法を実装して性能評価を行った．

3 章の評価と同様に，図 3 の実験環境にて Linux1-Linux2 間と，Windows-Linux2 間で iperf のコネクションを同時に確立し，通信速度を測定した．

提案手法における評価結果を図 6～図 9 に示す．図の p=100%は既存の CoDel と同等である．図 7 より提案手法が TCP 輻輳制御アルゴリズム間の公平性を大きく改善していることが確認できる．

図 9 のおける p と通信速度の関係に着目すると，p が大き過ぎると優先制御が不十分であり，不公平性が十分に解消できていないことがわかる．今回の測定の

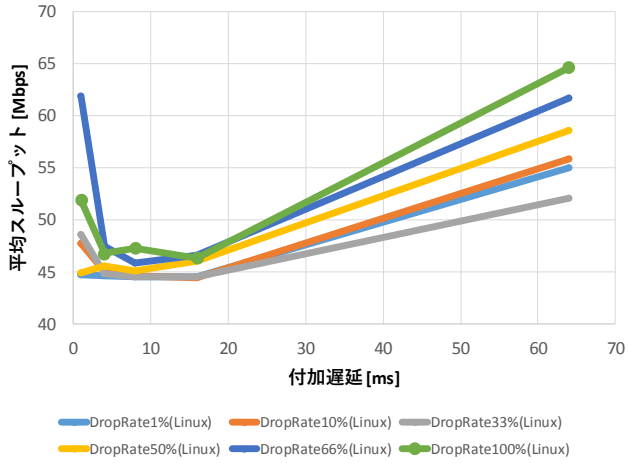

図 6　提案手法の評価結果(Linux)

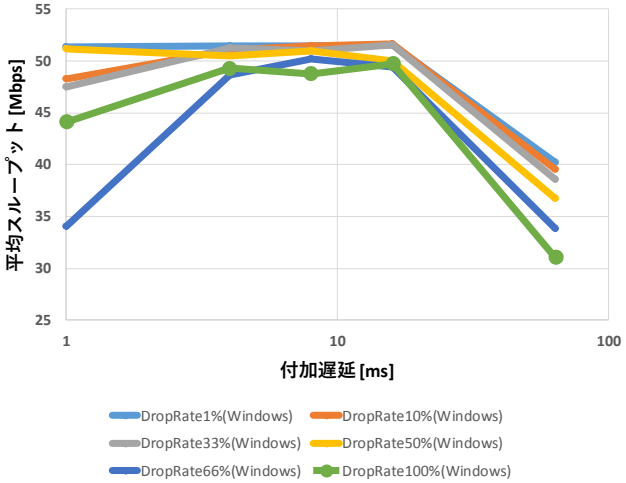

図 7　提案手法の評価結果(Windows)

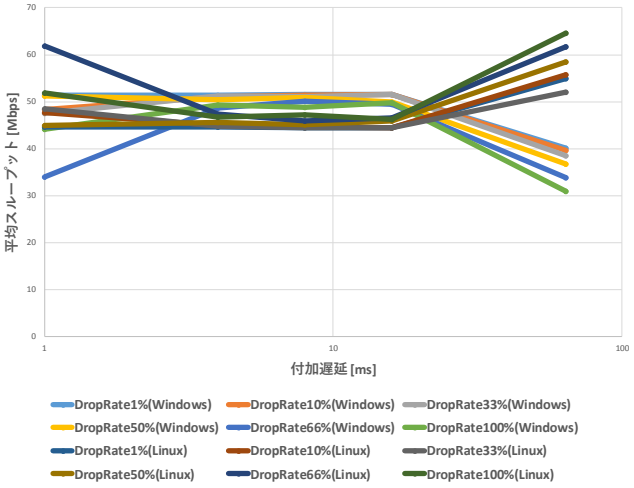

図 8　提案手法の評価結果

例では破棄率 33%程度が適切な値となっている．また p が過度に小さい場合も公平性が好ましくないことを

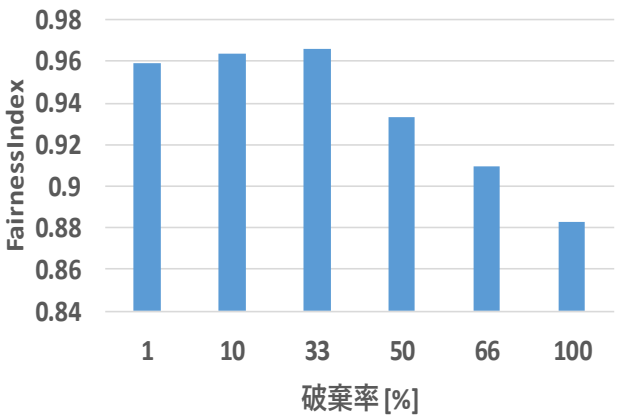

図 9　Fairness Index

確認できるが，p が小さすぎることによる負の影響は大きくなく，p は小さい値とすることが適切であることがわかる．

6. おわりに

本稿では既存の TCP 公平性向上手法としてネットワーク帯域を大きく消費する接続が既知であることを仮定し静的制御手法を紹介し，トラフィックの観察によりこの前提を要しない手法を提案した．そして，性能評価によりその有効性を示した．

今後はチューニングパラメータの決定に関する考察，遅延時間制御手法のパラメータ target を用いた公平性制御，既存の静的制御手法との比較について考察していく予定である．

謝辞

本研究は JSPS 科研費 25280022，26730040，15H02696 の助成を受けたものである。
本研究は、JST、CREST の支援を受けたものである。

文　献

[1] W. Richard Stevents, "TCSlow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," IETF RFC 2581, 1997.

[2] D.Katabi, M.Handley, and C.Rohrs, “Congestion control for high bandwidth-delay product networks,” in Proceedings of ACM SIG-COMM 2002,Aug.2002.

[3] 大浦亮, 山口実靖, "実機を用いた高速 TCP の公平性の評価", FIT2011 第 10 回情報科学技術フォーラム, RL-003, Sep. 2011

[4] Ryo Oura, Saneyasu Yamaguchi, “Fairness Comparisons Among Modern TCP Implementations,”The 6th International Workshopon Telecommunication Networking, Applications and Systems (TeNAS 2012), Mar. 2012.

[5] L. Xu, K. Harfoush and I. Rhee, “Binary Increase Congestion Control for Fast Long-Distance Networks,” Proc. IEEE Info COM 2004, March 2004

[6] Injong Rhee and LisongXu “CUBIC: A New TCP-Friendly High-Speed TCP Variant,” Proc. Workshop on Protocols for Fast Long Distance Networks, 2005.

[7] Kun Tan, Jingmin Song, Qian Zhang, and MurariSridharan,” A Compound TCP Approach for High-speed and Long Distance Networks” Proc.IEEE Info COM 2005,July 2005.

[8] 逸身勇人，山本幹，“CUBIC と Compound TCP 間の公平性改善手法の提案，” 電子情報通信学会信学技報 vol. 110, no. 372, NS2010-160，pp. 103-108

[9] Ryo Oura, Saneyasu Yamaguchi, “Fairness Analysis among Modern TCP Congestion Avoidance Algorithms Using Actual TCP Implementation and Actual Network Equipments,” ICNC 2011: 297-299

[10] 長谷川剛，板谷夏樹，村田正幸，“バックボーンルータにおける RED の動的閾値制御方式，” 電子情報通信学会信学技報 NS2001-11

[11] Ryo Oura, Saneyasu Yamaguchi, “Fairness Comparisons among Modern TCP Implementations,” AINA Workshops 2012: 909-914

[12] S. Floyd and V. Jacobson, “Random early detection gateways for congestion avoidance,” IEEE/ACM Transactions on Networking, vol. 1, pp. 397.413, Aug. 1993.

[13] Akiyama, Y.; Kozu, T.; Yamaguchi, S., “Active packet dropping for improving performance fairness among modern TCPs,” in Network Operations and Management Symposium (APNOMS), 2014 16th Asia-Pacific , vol., no., pp.1-4, 17-19 Sept. 2014

[14] Kathleen Nichols and Van Jacobson. 2012. Controlling queue delay.Commun. ACM 55, 7 (July 2012), 42-50.

[15] 花井雅人，山口実靖, "実機実装を用いた遅延時間制御手法の改良による TCP 公平性の向上", ネットワークシステム研究会(NS),(42)NS, March. 2016

[16] L. S. Brakmo and L. L. Peterson, “TCP Vegas: End to End Congestion Avoidance on a Global Internet”, IEEE Journal on Selected Areas in Communication, Vol.13, No.8, pp.1465-1480, October 1995.

[17] 秋山友理愛，大浦亮，神津智樹，山口実靖，“実機と実 TCP 実装を用いた TCP 公平性の評価” 電子情報通信学会 信学情報 NS2012-149，pp.49-54，2012

[18] Jim Gettys and Kathleen Nichols. 2011. Bufferbloat ACM (November 2011),

[19] iperf homepage, https://iperf.fr/