TCP BBR の RTprop の更新と TCP 公平性の改善に関する一考察

勝俣夏輝[†1,a)]　小川浩平[†1]　山口実靖[†1,b)]

概要：TCP BBR と CUBIC TCP が競合して通信を行うと，通信公平性が低くなることやスループットが周期的に大きく変動することが確認されている．また，TCP BBR に強制的に expire 処理を行わせることにより公平性が向上することや周期的スループット上下変動が緩和されることが過去の研究にて示されている．しかし，この手法は TCP BBR 単独通信時(非競合時)の TCP BBR の性能を低下させることが分かっている．本稿では，TCP BBR に expire 処理でなく RTprop 更新をさせたときの，競合通信時の性能公平性と単独通信時の性能を評価し，RTprop 更新をさせることの有効性について考察する．

1. はじめに

TCP は，OSI 参照モデルのトランスポート層に位置するプロトコルで，End-to-End 間のデータ転送における完全性を担保する他，経路上の輻輳を制御している．TCP 輻輳制御アルゴリズムとは，TCP に内蔵されているアルゴリズムで，通信時に用いるウィンドウサイズ(輻輳ウィンドウサイズ)を決定する．現在は TCP BBR [1]と CUBIC TCP[2]が大きなシェアを持ち，主として使用されている．TCP BBR は 2016 年に Neal Cardwell らによって提案された輻輳制御アルゴリズムである．TCP BBR は CUBIC TCP などの既存のロスベースの手法とは異なり，実際の経路上の輻輳状態に基づいてウィンドウサイズを決定する．具体的には，Kleinrock の輻輳モデル[3]に基づき，ウィンドウサイズを BDP (Bandwidth Delay Product)と等しくすることを目指す．TCP BBR はウィンドウサイズを BDP に等しくするために，通信をしながらボトルネック帯域幅と物理伝搬遅延時間を推定し，両者の積をウィンドウサイズとする．

TCP BBR と CUBIC TCP がボトルネックリンクを共有しながら同時に通信を行う際，CUBIC TCP が RTT を増加させ続けるため，TCP BBR と CUBIC TCP の間のスループットの公平性が非常に低くなることが知られている．この問題に対する解決策の一つとして，TCP BBR に強制的に *expire* 処理を行わせることで性能公平性を向上させる手法が提案されている [4]．しかし，この手法では TCP BBR が単体通信を行った際にスループットが低下するなどの課題がある．

本稿ではこの課題を解決や緩和するために，TCP BBR に対して *expire* 処理ではなく，*RTprop* の更新をさせる手法について考察する．具体的には，同手法の通信競合時における性能公平性と単体通信時における性能を示し，本手法の有効性について考察をする．

2. 関連研究

2.1 CUBIC TCP

CUBIC TCP [2]は Sangtae Ha らによって提案された TCP 輻輳制御アルゴリズムであり, Linux などの多くの OS において広く使用されている．スロースタートフェーズ時には自身の輻輳ウィンドウサイズを指数関数的に増加させ，輻輳回避フェーズ時には輻輳ウィンドウサイズを以下に示す三次関数の式に従って制御する．

$$cwnd = C(t-K)^3 \quad (1)$$

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

ここで，*cwnd* は輻輳ウィンドウサイズ，*t* は最後のパケットロスからの時間, W_{max}は最後のパケット損失時の輻輳ウィンドウサイズ, Cおよびβはそれぞれウィンドウサイズの増減速度を調整するためのパラメータである．C が大きいほど, 輻輳ウィンドウサイズの増加速度は高くなり, βが小さいほど，パケットロス検出時のスループット低下が小さくなる．図 1 に CUBIC TCP の輻輳ウィンドウサイズの推移の例を示す.

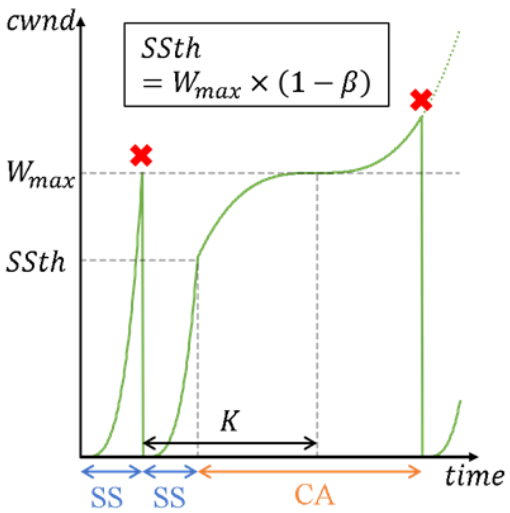
CUBIC TCP はロスベース手法の輻輳制御アルゴリズムであるため，パケットロスが頻繁に発生している環境ではスループットが著しく低下することが知られている. また，キュー長の増加やキューイング遅延時間の増加などの弊害を考慮せずに，パケットロスを検出するまで輻輳ウィンドウサイズを増加させているため，他の輻輳制御アルゴリズム(遅延ベース手法など)との相互作用を引き起こすことが知られている.

2.2 TCP BBR

TCP BBR [1]は，2016 年に Neal Cardwell らによって提案された輻輳制御アルゴリズムである. CUBIC TCP などのロスベース手法の輻輳制御アルゴリズムではパケットロスを検出するまで自身の輻輳ウィンドウサイズを増加させるため，常に通信経路上のルータにはキューが存在し，RTT は高い値を維持し続ける BufferBloat [5][6]が生じてしまうことが知られている．これに対して TCP BBR は，経路上のルータにキューを作成せず，BufferBloat を回避することを目指している.

†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, Shinjuku-ku, Tokyo, Japan.

a) cm21010@ns.kogakuin.ac.jp
b) sane@cc.kogakuin.ac.jp



SS:Slow Start phase

CA:Congestion Avoidance phase

図 1 CUBIC TCP のウィンドウサイズの推移

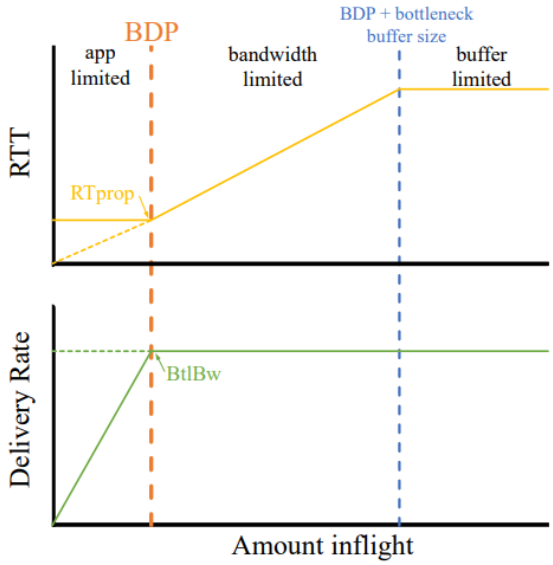


図 2 Kleinrock の輻輳モデル

TCP BBR は，図 2 に示す Kleinrock の輻輳モデルに基づいて通信経路のボトルネック帯域幅(*BtlBw*)と物理伝搬遅延(*RTprop*)を推定する．そして，推定された *BtlBw* と *RTprop* から BDP (Bandwidth-delay product)を算出し，自身のウィンドウサイズが算出された BDP になるよう制御する．BDP は，基本的に *BtlBw* と *RTprop* の積として求める．TCP BBR は既存の輻輳制御アルゴリズムとは異なり，パケットロスの検出や遅延時間増加の検出に直接は反応しない．

TCP BBR には 4 つのモードが存在する．図 3 に TCP BBR のモードとモード間の遷移を示す．通信開始直後は START_UP モードである．このモードでは，TCP BBR は輻輳ウィンドウサイズを指数関数的に増加させ，BDP を推定する．BDP を推定するためには，輻輳ウィンドウサイズが

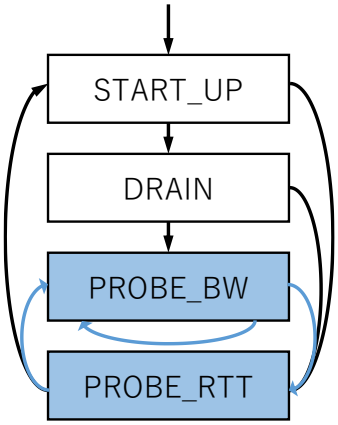


図 3 通信中の TCP BBR の Mode の推移

BDP を超える必要があるため，ボトルネックのバッファにキューを作ることになる．TCP BBR は輻輳ウィンドウサイズが BDP よりも大きくなったことを検出すると，DRAIN モードに移行する．このモードでは，輻輳ウィンドウサイズを減少させ，START_UP モード時に作成したキューを排出する．この結果，輻輳ウィンドウサイズは BDP と等しくなる．その後，PROBE_BW モードに移行する．このモードでは，pacing_gain と呼ばれるパラメータに従って，輻輳ウィンドウサイズを増減させることにより，利用可能な帯域(*BtlBw*)を探索する．PROBE_BW モードの開始から 10 秒が経過すると，TCP BBR は PROBE_RTT モードへ移行する．このモードでは，輻輳ウィンドウサイズを 4 セグメントにすることで，通信経路上のキューからパケットを排出することを試みる．次に，キューが空の状態(TCP BBR がキューが空であると期待している状況)で RTT を取得し，これを *RTprop* として記録する．その後，PROBE_BW モードへ移行し，通信終了まで PROBE_BW モードと PROBE_RTT モードを繰り返す．TCP BBR の実装ではこの，PROBE_BW モードの開始から 10 秒の経過を *expire* と呼んでいる．本稿では，*expire* 後に行われる PROBE_RTT モードへの移行と PROBE_BW モードへの移行を *expire* 処理と呼ぶ．*Expire* 処理には，輻輳ウィンドウサイズを 4 にして，*RTT* を計測して，*RTprop* を更新する処理が含まれている．

2.3 強制 expire による TCP BBR と CUBIC TCP の通信公平性向上手法

TCP BBR と CUBIC TCP がボトルネックリンクを共有しながら同時に通信を行うと，TCP BBR と CUBIC TCP のスループットが周期的に変動し[7][8]，両者の通信公平性が非常に低くなることが知られている [9][10]．この原因として，TCP BBR がキューが空の状況における遅延時間を *RTprop* として採用しても，共存する CUBIC TCP がキューを作成し拡大していき，実際に通信を行っている状況の遅延時間は TCP BBR が保持している *RTprop* よりも大きな値となってしまい，TCP BBR の輻輳ウィンドウサイズが実際の通信

状況の BDP に対して不適切に小さい値となってしまうことが示されている[7].

また，CUBIC TCP と TCP BBR が同時に通信を行うときに，TCP BBR が推定した *RTprop* と *RTT* が大きく乖離した場合に TCP BBR に強制的に *expire* 処理を行わせることで TCP 間スループット公平性が向上することが分かっている[4]．文献[4]では，TCP BBR において *RTT* / *RTprop* が閾値(*rate_th*)を超えた場合と，パケットロスを検出した場合に強制的に *expire* 処理を行わせて公平性の改善を行う手法を提案している．そして，性能評価により TCP 間スループット公平性が改善することを示している．この手法は，通信経路にキューがたまっている状況にて *RTT* を再度計測し，再測定された新しい *RTT* を *RTprop* として用いることにより，TCP BBR の *RTprop* を実際の通信状況に適したものに修正し，前述の TCP BBR の輻輳ウィンドウサイズが実際の通信状況の BDP に対して不適切に小さい値となってしまっている状況を解消や緩和をする手法であると言える．文献[4]の手法は，TCP BBR のスループットが CUBIC TCP のスループットと比較して著しく低い状況の緩和方法として，*RTT* / *RTprop* が閾値(*rate_th*)を超えたときに強制的に *expire* 処理をさせる手法を提案している．また，CUBIC TCP のスループットが TCP BBR のスループットと比較して著しく低い状況の緩和方法として，パケットロス検出時に強制的に *expire* 処理をさせる手法を提案している．本稿では，前者のみに着目し考察を行い，後者については考察を行わない．

図 4, 5 に，文献[4]で提案されている手法である強制 *expre* 手法を著者の勝俣らが実装して追実験をした結果を示す．図 4 はオリジナルの TCP BBR と CUBIC TCP のスループットの推移を示している．図 5 は，TCP BBR に対して閾値(*rate_th*)を 2.4 とし，強制的に *expire* 処理を行わせた際の TCP BBR と CUBIC TCP のスループットの推移を示している．また，強制 *expire* 手法を適用した TCP BBR が単体で通信した際のスループットも計測を行った．オリジナルの TCP BBR および強制 *expire* 手法を適用した TCP BBR の単体通信時のスループットを図 6 に示す．図内の“original bbr”がオリジナルの TCP BBR であり，“mod. bbr”が改変した(強制 *expire* を適用した)TCP BBR である．

図 4，図 5 から TCP BBR に対して強制的に *expire* 処理を行わせたことで両 TCP のスループットの周期的上下変動が緩和され，かつ通信公平性が向上したことが読み取れる．一方で，図 6 から，この手法を適用した TCP BBR が単体で通信を行う際にはスループットが低下してしまうことが分かる．

また文献[11]から，パラメータ *rate_th* の値の変動に対して，両者のスループット公平性は大きく変動せず，*rate_th* が 2 から 3 程度の場合に最も良いあるいはそれに近い公平性が得られることが分かっている．すなわち，本提案手法はチューニングパラメータである *rate_th* を最適化しなければ高い通信公平性を得られない手法ではないと言える．

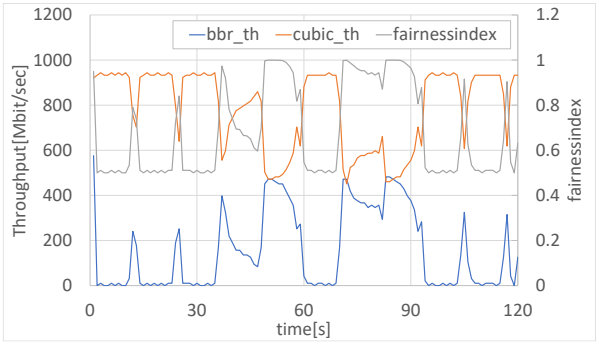


図 4 オリジナルの TCP BBR と CUBIC TCP のスループット推移

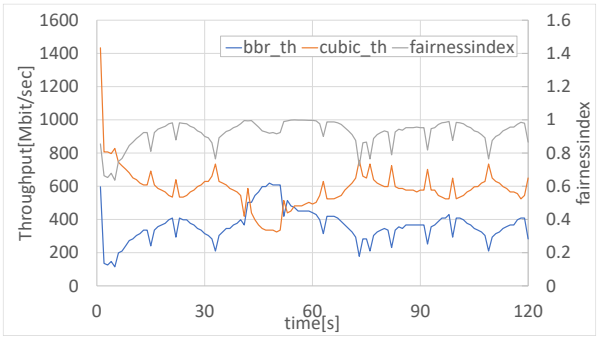


図 5 強制 expire を適用した TCP BBR と CUBIC TCP のスループット推移 (*rate_th*=2.4)

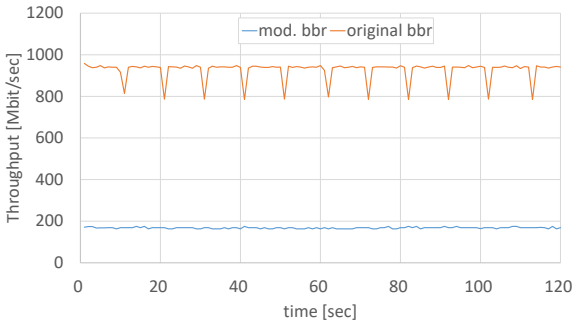


図 6 オリジナルおよび強制 expire を適用した TCP BBR による単体通信時のスループット

また，TCP BBR と CUBIC TCP がボトルネックリンクを共有しながら通信を行う際，TCP BBR の推定する物理伝搬遅延である *RTprop* の値が，物理伝搬遅延とキューイング遅延の両方を含む実際の通信経路の伝搬遅延である *RTT* よりも著しく小さくなってしまうこと，特に通信開始直後に小さくなってしまうこと，そしてこれにより通信開始直後の TCP BBR のスループットが著しく低下することが分かっている[4]. 強制的に *expire* 処理を行わせる手法では *expire* 処理によって TCP BBR が *RTprop* を再推定することでスループットの周期的上下変動および通信公平性が向上している．しかし，TCP BBR が *expire* 処理を行うと，*RTprop* を再

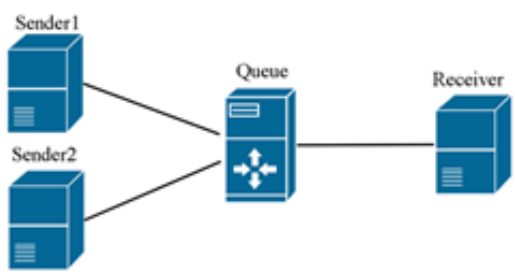


図 7 実験ネットワーク

表 1 計算機仕様

	Sender1, Sender2, Queue	Receiver
CPU	Intel Celeron CPU G530, 2.40GHz	Intel Celeron CPU G1101, 2.27GHz
Memory	16GB	8GB
OS	Linux-4.13.2	Linux-3.10.0

推定する際に一時的に輻輳ウィンドウサイズが 4 セグメントになるため，一時的にスループットが大幅に低下する．単体通信時などにて *expire* が頻繁に行われると TCP BBR のスループットが低下してしまうこととなる．

3. 提案手法

本章では TCP BBR に対して強制的に *expire* 処理を行わせるのではなく，*RTprop* の更新をさせることにより TCP 間のスループット公平性(主として TCP BBR と CUBIC TCP の間のスループットの公平性)の改善を行う手法を提案する．

提案手法の動作は以下の通りである．

起動条件： TCP BBR において *RTT* / *RTprop* が閾値(*rate_th*)を超えたときと，パケットロスを検出したときに，提案手法は起動され動作する．*RTT* / *RTprop* の算出は *RTT* の計測ごとに行われる．*rate_th* は 1< *rate_th* を満たすチューニングパラメータである．

起動時の動作： 起動時に測定されているの *RTT* と，起動時の *RTprop* の間の値を新しい *RTprop* として採用する．すなわち $(1-\alpha)\times RTT + \alpha \times RTprop$ を新しい *RTprop* として採用する．αは，$0 \leqq \alpha \leqq 1$ を満たすチューニングパラメータである．

上記の起動条件は，既存手法[4]の起動条件と同一である．起動時の動作は，既存手法とは異なる．既存手法では，輻輳ウィンドウサイズの 4 への縮小，*RTT* の再測定，再測定された最新の *RTT* の *RTprop* への適用という手順をとっているが，提案手法は測定済みの最新の *RTT* を *Rtprop* に適用することのみを行っている．よって，輻輳ウィンドウサイズの 4 への縮小による一時的な速度低下の発生を回避しており，単体通信時などにおける速度低下の課題を軽減できると期待される．

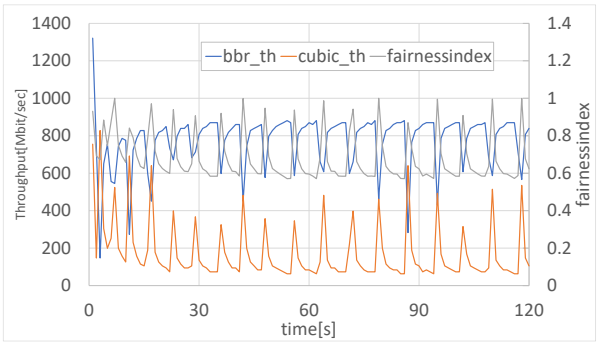


図 8 RTprop 書き換え手法によるスループット推移
(*rate_th*=2.0, α=1.0)

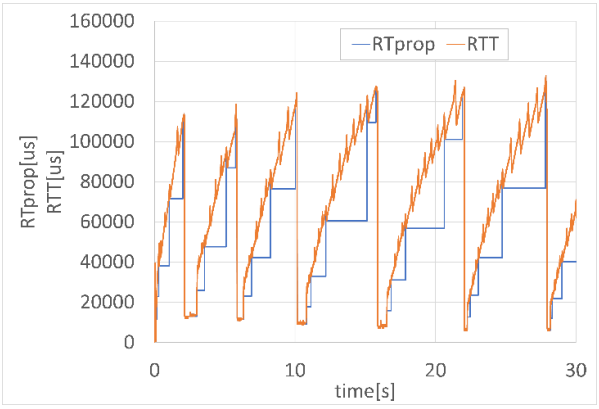


図 9 TCP BBR の *RTprop* および *RTT* の推移
(*rate_th*=2.0, α=1.0)

4. 性能評価

提案手法である *RTprop* 書き換え手法を実装し，スループットの測定を行った．実験は，図 7 の実験環境を構築して行った．Sender1 マシンからは TCP BBR で 1 コネクションを，Sender2 マシンからは CUBIC TCP で 1 コネクションを Receiver マシンに対して確立して通信を行い，スループットの測定を行った．スループットの計測は iperf 3.5 にて行った．これらは文献[4]と同一の実験環境である．使用した計算機の仕様は表 1 の通りである．

TCP BBR と CUBIC TCP のスループットの推移を図 8 に示す．また，通信中の TCP BBR の RTprop と RTT の推移を図 9 に示す．閾値(*rate_th*)は 2.0 と，αは 1 としている．既存研究[11]より，公平性は *rate_th* に強い影響を受けず，*rate_th* が 2.0 から 3.0 近辺で良い公平性が得られることが分かっており，本稿では 2.0 を用いた．

図 8 から，αを 1 とし，*RTT* 値をそのまま *RTprop* をとして採用した場合，TCP BBR の性能が大幅に上昇し，*RTprop* が不適切に小さいという課題は解決か大幅に緩和がされていることが分かる．一方で，TCP BBR のスループットが過

度に上昇し，通信公平性が既存手法と比較して低くなってしまっていることが分かる．図 9 より，*RTprop* は RTT に追従して急激に上昇していることが分かる．

次に，α を 0.5 にした場合の両 TCP のスループットの推移を図 10 に示す．閾値(*rate_th*)は同様に 2.0 とした．また，通信中の TCP BBR の *RTprop* および *RTT* の推移を図 11 に，単体通信を行った際のスループットを図 12 に示す．

図 10，図 11 から，α を 1 として *RTT* を *RTprop* に対してそのまま適用した場合と比較して，通信中の TCP BBR の *RTprop* の増加が抑えられ，TCP BBR および CUBIC TCP の通信公平性が向上していることが分かる．また，図 12 から提案手法を適用した TCP BBR の単体通信時のスループットは，強制 *expire* 手法を適用した TCP BBR のものとは異なり，著しい低下が発生していないことが分かる．

これらの結果から，TCP BBR に対して *RTT* / *RTprop* が閾値(*rate_th*)を超えた際に *RTprop* を更新することによって CUBIC TCP とボトルネックリンクを共有して通信を行った際の通信公平性が向上し，強制 *expire* 手法で低い性能が見られた単体通信時でも高いスループットを得ることができることが確認できた．換言すると，既存手法[4]と近い公平性改善の利点が得られ，既存手法の単体通信時の性能低下の欠点を大幅に縮小できたことが確認できた．ただし，チューニングパラメータの影響も確認でき，この最適は重要な課題であると考える．

5. 考察

TCP BBR と CUBIC TCP がボトルネックリンクを共有して通信を行う際に通信公平性を向上させる手法として，強制的に *expire* 処理を行わせる手法が提案されている[4]．また，強制 *expire* 手法の課題として，閾値(*rate_th*)が低く *expire* 処理が頻発する状況下では TCP BBR のスループットが低くなってしまう点があげられる．この欠点のナイーブな解決手法としては，閾値(*rate_th*)を大きくして強制 *expire* 手法が起動しづらくする方法が考えられる．しかし，強制 *expire* 手法の起動数を減らす方法は欠点(一時的性能低下)の影響を減らすだけでなく，利点(TCP BBR の性能が不適切に低い状況の解決)の影響も減らし，過度に大きな閾値(*rate_th*)は無改変の実装と実質的に同等となってしまう．そして，閾値(*rate_th*)の設定の一般化は難しく，この調整のみにより欠点の解決を行うことは困難であると思われる．今回我々が提案した手法は TCP BBR の一時的な性能低下の欠点を伴わない分，TCP BBR の性能が高くなる可能性がある．そして，パラメータ α の調整により TCP BBR の性能劣化の解決(性能向上)を抑制的にしている．既存手法と提案手法を比較すると，既存手法は，処理の利点と欠点が大きく，その処理を抑制的に実行する必要があり，提案手法は，処理の欠点が小さく，高い頻度で実行することができるものである．これにより制御がより細かい粒度で適切に行えると期待できる．

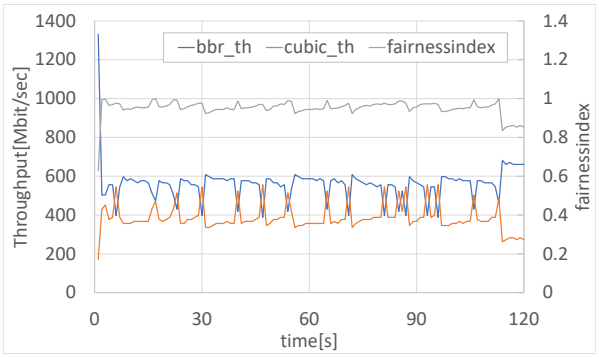


図 10 RTprop 書き換え手法によるスループット推移 (*rate_th*=2.0, α=0.5)

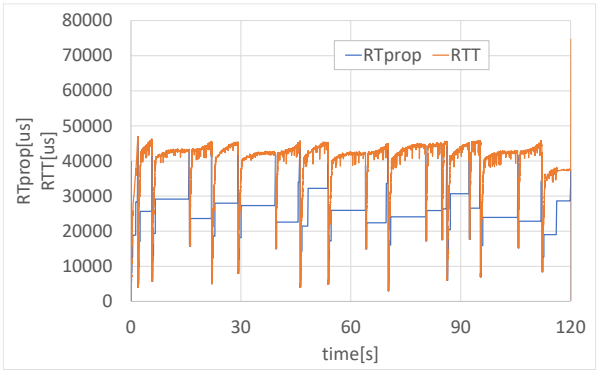


図 11 TCP BBR の *RTprop* および *RTT* の推移 (*rate_th*=2.0, α=0.5)

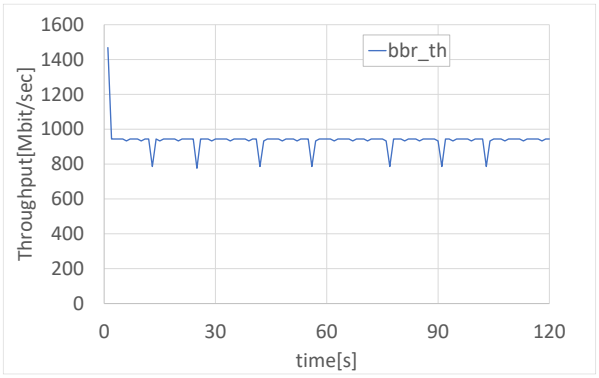


図 12 提案手法を適用した TCP BBR による単体通信時のスループット推移 (*rate_th*=2.0, α=0.5)

また，TCP BBR はキューを作らないことを目的としており，CUBIC TCP がキューを作っている状況にてそれを含めた大きさの *RTT* を *RTprop* として採用することは，TCP BBR の本来に趣旨に適さないと考えることもできる．一方で，TCP BBR が他の TCP 輻輳制御アルゴリズムと共存時に性能公平性が実現できなかったり，性能が著しく低くなってしまったりする課題の多くが，TCP BBR はキューを作らないことを目指して動作するが，他の既存のアルゴリズムが TCP BBR の趣旨を無視してキューを作成してしまうことによる[7]．よって，他の TCP アルゴリズムが存在している

状況にて TCP BBR が普及をしていくためには，*RTprop* と実 *RTT* の乖離の課題は解決する必要があると考えられる．本稿で提案した，*RTprop* と実 *RTT* の中間の値を新しい *RTprop* として採用する手法は，TCP BBR の趣旨を残しつつ，課題の解決も行う手法であると考えられる．また，我々は今後は TCP BBR の普及が進むことを期待している．よって，初期は α を大きめにして他の TCP 輻輳制御アルゴリズムとの公平性の考慮を大きくし，TCP BBR 普及の後は，α を小さくし，TCP BBR の趣旨であるキューを作らずに Bufferbloat を回避することの考慮を大きくするなどの運用が考えられる．

RTprop の更新による *RTprop* と実 *RTT* の乖離の課題の解決により，単体通信時にはキューを作り RTT を増加させる可能性がある．ただし，本稿で提案した手法は頻繁に *RTprop* の更新を行わず，*RTprop* の拡大も抑制的であるため RTT の増加も限定的なものになると期待できる．

6. まとめ

TCP BBR と CUBIC TCP がボトルネックリンクを共有した状況下で通信を行った際に，通信公平性が著しく低くなることが分かっている．この問題に対する解決案として TCP BBR に対して *RTT* / *RTprop* が閾値(*rate_th*)を超えた場合に強制的に *expire* の処理を行わせることで通信公平性が向上することが過去の研究によって示されている．しかし，この手法では単体通信時にスループットが低下してしまう問題が挙げられている．本稿では，強制的に *expire* 処理を行うのではなく，*RTprop* を書き換える手法を提案した．そして性能評価により，CUBIC TCP との競合時の公平性を担保しつつ，単体通信時にもスループットの低下が発生しない状況を実現できることを示した．

今後は，チューニングパラメータの設定方法についての考察，状況の一般化についての考察，他の TCP アルゴリズムでの評価を行っていく予定である．

参考文献

[1] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson, “BBR: Congestion-Based Congestion Control,” Queue 14, 5, pages 50 (October 2016), 34 pages, 2016. DOI: https://doi.org/10.1145/3012426.3022184

[2] Injong Rhee and Lisong Xu “CUBIC: A new TCP-friendly high-speed TCP variant,” Proc. Workshop on Protocols for Fast Long Distance Networks, 2005.

[3] Kleinrock, L, “Power and deterministic rules of thumb for probabilistic problems in computer communications,” In: ICC '79; International Conference on Communications, Boston, Mass., June 10-14, 1979, Conference Record. vol. 3. (A80-25901 09-32) Piscataway, N.J., Institute of Electrical and Electronics Engineers, Inc., 1979, pp. 43.1.1-43.1.10.

[4] K. Sasaki, K. Miyazawa, K. Ogawa, S. Yamaguchi, “TCP BBR Performance Improvement on a Network with Increasing RTT,” The Eighth International Symposium on Computing and Networking (CANDAR 2020), Nov. 2020.

[5] J. Gettys, "Bufferbloat: Dark buffers in the internet," in IEEE Internet Computing, vol. 15, no. 3, pp. 96-96, May-June 2011.Kathleen Nichols and Van Jacobson. 2012. Controlling queue delay. Commun. ACM vol. 55(7) (July 2012), pp. 42-50.

[6] Vint Cerf, Van Jacobson, Nick Weaver, and Jim Gettys. 2011. BufferBloat: what's wrong with the internet?. Queue 9, 12, Pages 10 (December 2011), 11 pages. DOI: https://doi.org/10.1145/2076796.2076798

[7] K. Miyazawa, K. Sasaki, N. Oda and S. Yamaguchi, "Cycle and Divergence of Performance on TCP BBR," 2018 IEEE 7th International Conference on Cloud Networking (CloudNet), 2018, pp. 1-6, doi: 10.1109/CloudNet.2018.8549411.

[8] K. Miyazawa, K. Sasaki, N. Oda and S. Yamaguchi, "Cyclic Performance Fluctuation of TCP BBR," 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), 2018, pp. 811-812, doi: 10.1109/COMPSAC.2018.00132.

[9] M. Hock, R. Bless and M. Zitterbart, "Experimental evaluation of BBR congestion control," 2017 IEEE 25th International Conference on Network Protocols (ICNP), Toronto, ON, 2017, pp. 1-10. doi: 10.1109/ICNP.2017.8117540

[10] Feng Li, Jae Won Chung, Xiaoxiao Jiang, and Mark Claypool,“TCP CUBIC versus BBR on the Highway,”In Proceedings of the Passive and Active Measurement Conference (PAM), March 26-27, 2018. doi: 10.1007/978-3-319-76481-8_20

[11] K. Miyazawa, N. Katsumata, S. Yamaguchi, “TCP Congestion Control outside Kernel,” IEICE 2020 International Conference on Emerging Technologies for Communications (ICETC 2020)