

# 異種輻輳制御機構の競合時における性能評価： CUBIC vs Copa vs BBR

荻野 雅史<sup>1</sup> 岡田 章吾<sup>1</sup> 内海 哲史<sup>2</sup>

概要：インターネットアプリケーションの多様化に伴い，端末間通信は高スループット，低遅延がますます求められるようになってきた．一方，ネットワーク機器に搭載されるバッファメモリのサイズの増加と，ロススペース輻輳制御機構の利用に起因し，バッファリング遅延が増大する現象である Bufferbloat が問題視されている．バッファリング遅延を抑える輻輳制御機構として，BBR が Google 社より発表され，のちに，MIT コンピュータサイエンス・人工知能研究所により Copa が発表された．本稿では，TCP 輻輳制御機構としてもっともシェアが高い，ロススペース輻輳制御機構である CUBIC と，バッファリング遅延を抑える最新の輻輳制御機構である Copa との競合時，及び，バッファリング遅延を抑える輻輳制御機構同士である Copa と BBR の競合時における性能評価を行う．

## Performance Evaluation of Heterogeneous Congestion Control Mechanisms: CUBIC vs Copa vs BBR

### 1. はじめに

メモリの低価格化やネットワーク機器の処理能力向上を背景として，ルーターやスイッチなどの交換機に搭載されるバッファメモリのサイズが過剰に搭載されることがある．従来から広く利用されているロススペース輻輳制御機構は，ボトルネックリンクのバッファを埋め尽くすまでパケット送信レートを増加させていくため，過剰にバッファが搭載された交換機において，バッファリング遅延が大きくなる問題を引き起こしている．このような現象は Bufferbloat [1] と呼ばれ，近年，インターネットにおいて観測されている．

Bufferbloat を回避する手段として，バッファリング遅延を抑える輻輳制御機構である BBR (Bottleneck Bandwidth and Round-trip propagation time) [2] が開発された．BBR は 2016 年，Google 社によって発表された輻輳制御機構である．米国のインターネットトラフィックの多くを占める YouTube をはじめとする，Google 社のサービスにおける輻輳制御機構として導入されている BBR は，今日のイン

ターネット環境に大きな影響を与えている．また，Copa [3] は 2018 年，MIT コンピュータサイエンス・人工知能研究所から発表された，バッファリング遅延を抑える最新の輻輳制御機構である．Copa は，伝搬遅延が大きい衛星ネットワーク環境においてもバッファリング遅延を抑え，リアルタイム通信を実現できる輻輳制御機構として期待される．Facebook 社は，Android プラットフォーム上の Facebook Live において，QUIC [4] に実装された Copa の性能評価を行い [3]，また，現在，Android 端末への動画配信に Copa を使用している [5]．一方，近年，Windows, macOS, Linux, Android OS におけるデフォルトの TCP 輻輳制御機構として，ロススペース輻輳制御機構である CUBIC [6] が用いられている．インターネットにおける CUBIC の TCP 輻輳制御機構としてのシェアはもっとも高く，36%となっている [7]．

BBR などのバッファリング遅延を抑える輻輳制御機構と CUBIC などのロススペース輻輳制御機構が競合するとき，ボトルネックリンクにおけるバッファサイズによって，スループットが不公平になることが知られている [8] [9]．特に，ボトルネックリンクにおけるバッファサイズが大きい場合，ロススペース輻輳制御機構である CUBIC と，バッファリング遅延を抑える輻輳制御機構である BBR が競合

<sup>1</sup> 福島大学大学院 共生システム理工学研究科, Graduate School of Symbiotic Systems Science and Technology, Fukushima University

<sup>2</sup> 福島大学 理工学群 共生システム理工学類, Faculty of Symbiotic Systems Science, Cluster of Science and Technology, Fukushima University

するとき、BBRのスループットが低下する。

CUBIC が広く使われている現在のインターネットにおいて、Bufferbloat を回避するためにバッファリング遅延を抑える輻輳制御機構を普及させていくには、CUBIC とバッファリング遅延を抑える輻輳制御機構の競合時におけるスループット公平性が重要である。また、現在シェアの高い CUBIC と BBR に対して、バッファリング遅延を抑える最新の輻輳制御機構として期待される Copa が与える影響は、あまり知られていない。そこで、本稿では、ネットワークエミュレータ Mininet [10] を用いた実験により、Copa と CUBIC が競合するとき、及び Copa と BBR が競合するときの性能評価を行う。

本稿の構成は以下の通りである。2章で関連研究について述べる。3章では、本稿で扱う TCP 輻輳制御機構のアルゴリズムについて述べる。4章では、本稿における性能の評価方法について述べる。5章で Copa と CUBIC の競合時の性能評価、6章で Copa と BBR の競合時の性能評価を行う。最後に、7章で結論と今後の課題を述べる。

## 2. 関連研究

文献 [8] では、ロススペース輻輳制御機構と競合するときの BBR の挙動に関する解析モデルを構築している。複数フローでボトルネックリンクを共有するとき、BBR は輻輳ウィンドウサイズによって送信中パケット数を制限する。この解析モデルにより、BBR フロー群の送信中パケット数の合計を求めることで、CUBIC フローや TCP Reno フローなどのロススペース輻輳制御機構のフローと競合するときにおける、BBR フロー群が占有する帯域の割合が予測可能であることを示している。文献 [9] では、ネットワークエミュレータ Mininet を用いて、BBR フローと CUBIC フローの競合時において、往復伝搬遅延時間やボトルネックリンクのバッファサイズを変化させ、スループット公平性を検証している。その結果、スループット公平性はボトルネックリンクのバッファサイズに大きく依存し、往復伝搬遅延時間の大きさにほとんど依存しないことを示している。また、BBR フローと CUBIC フローの数によらず、BBR フロー群は、少なくともボトルネックリンク帯域の 35% 以上を占有できると結論付けている。また、Copa を提案している文献 [3] では、Copa と CUBIC との競合時のスループットについて、ボトルネックリンクにおけるバッファサイズが比較的小さい場合 (0.5~5 [BDP]) において、BBR と PCC [11] よりも CUBIC に対して公平になると主張している。上述の関連研究では、BBR とロススペース輻輳制御機構との競合時の性能検証や、ボトルネックリンクにおけるバッファサイズが比較的小さいときの Copa と CUBIC との競合時の性能評価を行っている。本研究では、ボトルネックリンクにおけるバッファサイズが比較的大きい場合 (8~64 [BDP]) についても、Copa と CUBIC とが

競合するときの性能評価を行う。また、バッファリング遅延を抑える輻輳制御機構同士である最新の輻輳制御機構である Copa と、シェアの高い BBR が競合するときの性能についても検証する。

## 3. TCP 輻輳制御機構のアルゴリズム

本章では、本稿で扱う TCP 輻輳制御機構のアルゴリズムである、CUBIC, Copa, 及び BBR について説明する。

### 3.1 CUBIC

本節では、文献 [6] と、Linux4.5.18 及び最新カーネル Linux5.13 の実装に基づいて、CUBIC のアルゴリズムを説明する。CUBIC は、Windows, macOS, Linux, Android OS におけるデフォルトの TCP 輻輳制御機構であり、現在最も普及している TCP 輻輳制御機構である [7]。CUBIC で用いられるウィンドウ成長関数  $w(t)$ [packets] を式 (1) に示す。

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

ここで、 $W_{\max}$  [packets] はパケットロスを検出した時点の輻輳ウィンドウサイズに基づいて計算される。 $C$  はスケールリングファクタであり、 $t$  [sec] は輻輳回避開始からの経過時間である。具体的には、 $C = 0.4$  である。また  $K$  [sec] は輻輳ウィンドウサイズの増加速度を決定するパラメータであり、以下の式 (2) で計算される。

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

ここで、 $\beta$  はパケットロス検出時の輻輳ウィンドウサイズの減少率を表すパラメータである。具体的には、 $\beta = 0.3$  である。CUBIC は輻輳回避開始からの経過時間  $t$  に基づいて、三次関数的に輻輳ウィンドウサイズを増加させる。これによって広帯域のリンクにおいても高いスループットを実現することができる。また、輻輳ウィンドウサイズの増加量は輻輳回避開始からの経過時間  $t$  に関する関数であり、往復遅延時間 (RTT) に依存しない。

### 3.2 Copa

本節では、文献 [3] と CCP (Congestion Control Plane) [12] の実装に基づいて、Copa のアルゴリズムを説明する。Copa では、平均スループット  $\lambda$  [packets/sec] と、バッファリング遅延  $d$  [sec] からなる目的関数  $U$  を最大化することを目指す。目的関数  $U$  を式 (3) に示す。

$$U = \log(\lambda) - \delta \cdot \log(d) \quad (3)$$

ここで、 $\delta$  はスループットと比較してバッファリング遅延を重み付けするパラメータである。 $U$  を最大化する目標レート  $\lambda_t$  [packets/sec] を式 (4) に示す。

$$\lambda_t = \frac{1}{\delta \cdot d_q} \quad (4)$$

ここで,  $d_q$  [sec] は平均のバッファリング遅延の推定値である.  $d_q$  として, 式 (5) で算出される値を用いる.

$$d_q = RTT_{standing} - RTT_{min} \quad (5)$$

$RTT_{standing}$  [sec] は直近の過去の時間  $\tau$  [sec] で観測された最小の RTT である. 具体的には,  $\tau = sr_{tt}/2$  であり, ここで  $sr_{tt}$  [sec] は RTT の移動平均である.  $RTT_{min}$  [sec] は直近の過去  $W_{R1,Copa}$  [sec] の間に観測された最小の RTT である. 具体的には,  $W_{R1,Copa} = 10$  [sec] である.

送信側は確認応答を受信するごとに, 現在の平均スループット  $\lambda$  を推定する. 輻輳ウィンドウサイズを  $cwnd$  [packets] として, 式 (6) によって  $\lambda$  を算出する.

$$\lambda = cwnd/RTT_{standing} \quad (6)$$

式 (4) で求めた目標レート  $\lambda_t$  と式 (6) で求めた現在の平均スループットの推定値  $\lambda$  を比較して, 確認応答受信ごとに,  $cwnd$  を以下の式 (7), (8) で更新する.

$\lambda = cwnd/RTT_{standing} \leq \lambda_t$  の場合:

$$cwnd \leftarrow cwnd + v/(\delta \cdot cwnd) \quad (7)$$

$\lambda = cwnd/RTT_{standing} > \lambda_t$  の場合:

$$cwnd \leftarrow cwnd - v/(\delta \cdot cwnd) \quad (8)$$

ここで  $v$  は速度パラメータである. 速度パラメータ  $v$  は, 収束を加速するためのもので, 具体的には,  $v = 1$ , または  $v = 2$  が用いられる.

また, Copa には 2 つのモードが実装されている.  $\delta$  の値が固定のデフォルトモードと, バッファを埋め尽くすロスベース輻輳制御機構と競合する場合に,  $\delta$  の値を動的に調整する競合モードである. デフォルトモードでは,  $\delta = 0.5$  を用いる. バッファが直近の過去  $W_{R2,Copa}$  [sec] の間に空である状態を検出した場合<sup>(注1)</sup>, デフォルトモードを継続する. 具体的には,  $W_{R2,Copa} = 5 \cdot RTT$  [sec] である. また,  $W_{R2,Copa}$  の間にバッファが空である状態を検出なかった場合, 競合モードに移行する. 競合モードでは, パケットの到着やロスに応じて  $1/\delta$  を AIMD (Additive-Increase/Multiplicative-Decrease) で変化させる.

### 3.3 BBR

本節では, 文献 [2] と, Linux4.5.18 及び最新カーネル Linux5.13 の実装に基づいて, BBR のアルゴリズムを説明する. BBR では, 式 (9), (10) によって, 時刻  $T$  [sec] における端末間の往復伝搬遅延時間の推定値  $\widehat{RTprop}_T$  [sec] とボトルネックリンク帯域の推定値  $\widehat{BtlBw}_T$  [packets/sec] を求める. BBR は, 転送レートを  $\widehat{BtlBw}_T$  に到達させ,

<sup>(注1)</sup>Copa は, 直近の過去に観測した最大値の 10%未満のバッファリング遅延を観測したとき, ボトルネックリンクのバッファが空であると推測する.

かつ端末間の往復遅延時間  $RTT_T$  [sec] が  $\widehat{RTprop}_T$  となることを目指す. ここで,  $RTprop$  [sec] は端末間の往復伝搬遅延時間である.  $RTT_T$  を以下の式 (11) に示す.

$$\begin{aligned} \widehat{RTprop}_T &= RTprop + \min(\eta_t) \\ &= \min(RTT_t) \quad \forall t \in [T - W_{R,BBR}, T] \end{aligned} \quad (9)$$

$$\widehat{BtlBw}_T = \max(deliveryRate_t) \quad \forall t \in [T - W_{B,BBR}, T] \quad (10)$$

$$RTT_T = RTprop + \eta_T \quad (11)$$

ここで,  $\eta_T \geq 0$  であり,  $\eta_T$  [sec] はバッファリング遅延と各ネットワーク機器の処理遅延時間の合計を表している.  $W_{R,BBR}$  [sec] は時間ウィンドウであり, 具体的には,  $W_{R,BBR} = 10$  [sec] である.  $deliveryRate_T$  [packets/sec] はパケット送信から確認応答受信までの平均転送レートであり, 確認応答の受信間隔  $\Delta t_T$  [sec] に対する転送データ量  $\Delta delivered_T$  [packets] の大きさである. つまり, 平均転送レート  $deliveryRate_T$  を式 (12) で求める. また,  $W_{B,BBR}$  [sec] は時間ウィンドウであり, 具体的には,  $W_{B,BBR} = 10 \cdot RTT_T$  [sec] である.

$$deliveryRate_T = \Delta delivered_T / \Delta t_T \quad (12)$$

BBR では, コネクション開始時に Startup と Drain と呼ばれるモードが実行され, その後, ProbeBW と ProbeRTT と呼ばれるモードが交互に実行される.

## 4. 評価方法

本章では, 本稿における性能評価の実験環境と評価指標について述べる.

### 4.1 実験環境

図 1 に, Mininet によるエミュレーション実験で用いるネットワークポロジを示す. ネットワークは,  $2 \times n$  台の送信端末と  $2 \times n$  台の受信端末, 2 台の交換機,  $4 \times n$  本のアクセスリンク, 及び 1 本のボトルネックリンクから構成される. 送受信端末には, CCP がインストールされた Linux4.15.18 が搭載されている. 送信端末  $S_{1,1} \sim S_{1,n}$  の TCP 輻輳制御機構として Copa を, 送信端末  $S_{2,1} \sim S_{2,n}$  の TCP 輻輳制御機構として CUBIC, または BBR を用いる. また, 受信端末には, SACK (Selective Acknowledgement) オプション機能, タイムスタンプオプション機能, Delayed ACK (Delayed Acknowledgement) 機能を備えた受信側の TCP 輻輳制御機構を搭載する. ボトルネックリンクの帯域を 50 [Mbps], 端末間の往復伝搬遅延時間を 40 [msec], エミュレーション実験時間をそれぞれ 120 [sec] とする. アクセスリンク, ボトルネックリンクはそれぞれ有線リンクを想定し, エラーによるパケットロス率を 0% とする. まず,  $n = 1$  本, または,  $n = 4$  本として, Copa フローと CUBIC フローをボト

ルネックリンクで競合させ、ボトルネックリンクにおけるバッファサイズを  $0.5 \sim 64$  [BDP] で変化させる。ここで、 $1$  [BDP] =  $50$  [Mbps]  $\times$   $40$  [msec] /  $8$  [bits] =  $250$  [Kbytes] である。同様に、 $n = 1$  本、または、 $n = 4$  本として、Copa フローと BBR フローをボトルネックリンクで競合させ、ボトルネックリンクにおけるバッファサイズを  $0.5 \sim 64$  [BDP] で変化させる。

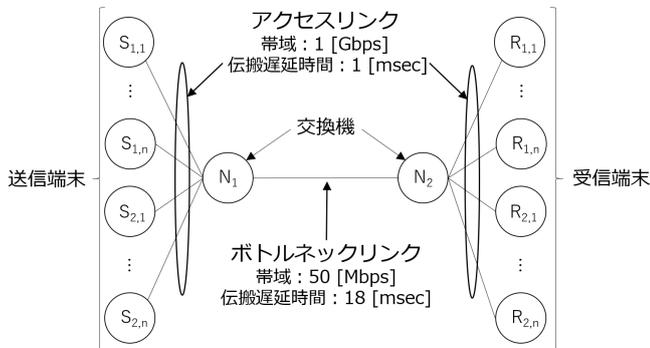


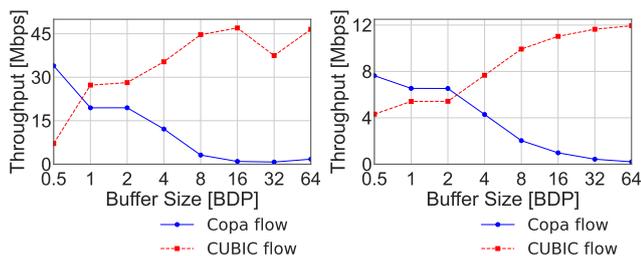
図 1: ネットワークポロジ

#### 4.2 評価指標

性能評価において用いる指標は、1 フローあたりの平均スループット、送信中パケット数の時間変化、及び RTT の時間変化である。これらの指標は、Tcpdump [13] によってキャプチャしたパケットデータを、tcptrace [14] によって解析して取得する。

### 5. Copa vs CUBIC

本章では、送信端末  $S_{1,1} \sim S_{1,n}$  における TCP 輻輳制御機構が Copa、送信端末  $S_{2,1} \sim S_{2,n}$  における TCP 輻輳制御機構が CUBIC のときの性能評価の結果について述べる。



(a)  $n=1$  本ずつ競合<sup>(注2)</sup>

(b)  $n=4$  本ずつ競合

図 2: バッファサイズに対する 1 フローあたりの平均スループットの変化 (Copa vs CUBIC)

図 2(a) に Copa フローと CUBIC フローが  $n = 1$  本ずつ競合したとき、図 2(b) に  $n = 4$  本ずつ競合したときのそれぞれについて、バッファサイズに対する 1 フローあたりの平均スループットの変化を示す。図 3 に Copa フ

ローと CUBIC フローが  $n = 1$  本ずつ競合したときにおける、Copa フローの送信中パケット数の時間変化を、図 4 に CUBIC フローの送信中パケット数の時間変化をそれぞれ示す。図 5 に Copa フローと CUBIC フローが  $n = 1$  本ずつ競合したときにおける Copa フローの RTT の時間変化を示す。

図 2(a) より、 $n = 1$  本ずつ競合したとき、バッファサイズが  $0.5$  [BDP] の場合、Copa フローのスループットが高く、バッファサイズが大きくなるにしたがって、CUBIC フローのスループットが高くなる傾向を示すことがわかる。

図 5 より、バッファサイズが  $0.5$  [BDP] の場合、RTT が大きくなっていないことがわかる。このとき、バッファリング遅延が小さいことから、Copa はバッファが空になることを検出するため、デフォルトモードで動作する割合が高くなっている。また、図 3, 4 より、バッファサイズが  $0.5$  [BDP] の場合、Copa フローの送信中パケット数が、CUBIC フローの送信中パケット数よりも大きくなっていることがわかる。Copa のデフォルトモードでの輻輳ウィンドウサイズは、パケットロスに依存しない。それに対して CUBIC は、パケットロスを検出すると輻輳ウィンドウサイズを小さくする。そのため、Copa フローのパケット送信は、CUBIC フローのパケット送信よりも積極的となり、Copa フローのスループットが高くなる。

図 5 より、バッファサイズが大きくなるにしたがい、RTT も大きくなる傾向を示すことがわかる。このとき、バッファリング遅延も大きくなり、バッファが空になることを検出しなくなるため、Copa は競合モードで動作する。また、図 3, 4 より、バッファサイズが大きくなるにしたがい、Copa フローの送信中パケット数に対して、CUBIC フローの送信中パケット数が大きくなっていることがわかる。Copa の競合モードでは  $1/\delta$  を AIMD で変化させる。それに対して CUBIC では、ウィンドウサイズを 3 次関数的に増加させる。そのため、CUBIC フローのパケット送信が Copa フローよりも積極的となり、Copa フローを圧迫し、Copa フローのスループットが低下する。バッファサイズが大きくなるにしたがい、CUBIC フローの送信中パケット数も大きくなり、CUBIC フローによってバッファが埋め尽くされることで、RTT が大きくなる。

図 2(b) より、 $n = 4$  本ずつ競合したとき、バッファサイズが  $2$  [BDP] 以下の場合、Copa フローのスループットのほうが大きく、バッファサイズが大きくなるにしたがって、CUBIC フローのスループットが大きくなる傾向を示すことがわかる。

このように、ロスベース輻輳制御機構である CUBIC とバッファリング遅延を抑える輻輳制御機構である Copa が

<sup>(注2)</sup>  $32$  [BDP] のとき、合計スループットが低下した原因は、再送タイムアウトによる。また、 $64$  [BDP] のとき、CUBIC の輻輳ウィンドウサイズは、その上限に達した。

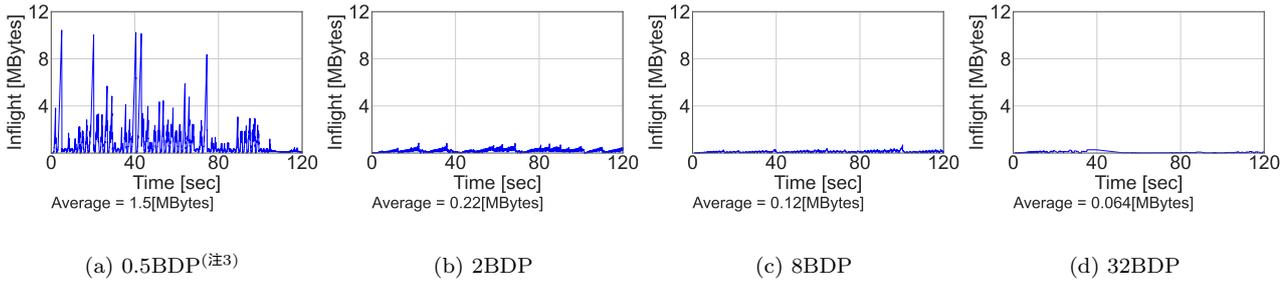


図 3: Copa フローの送信中パケット数の時間変化 (Copa フローと CUBIC フローが  $n = 1$  本ずつ競合)

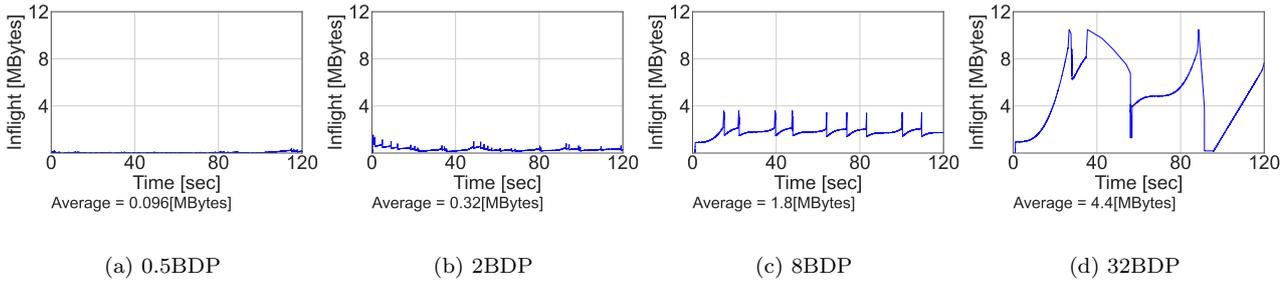


図 4: CUBIC フローの送信中パケット数の時間変化 (Copa フローと CUBIC フローが  $n = 1$  本ずつ競合)

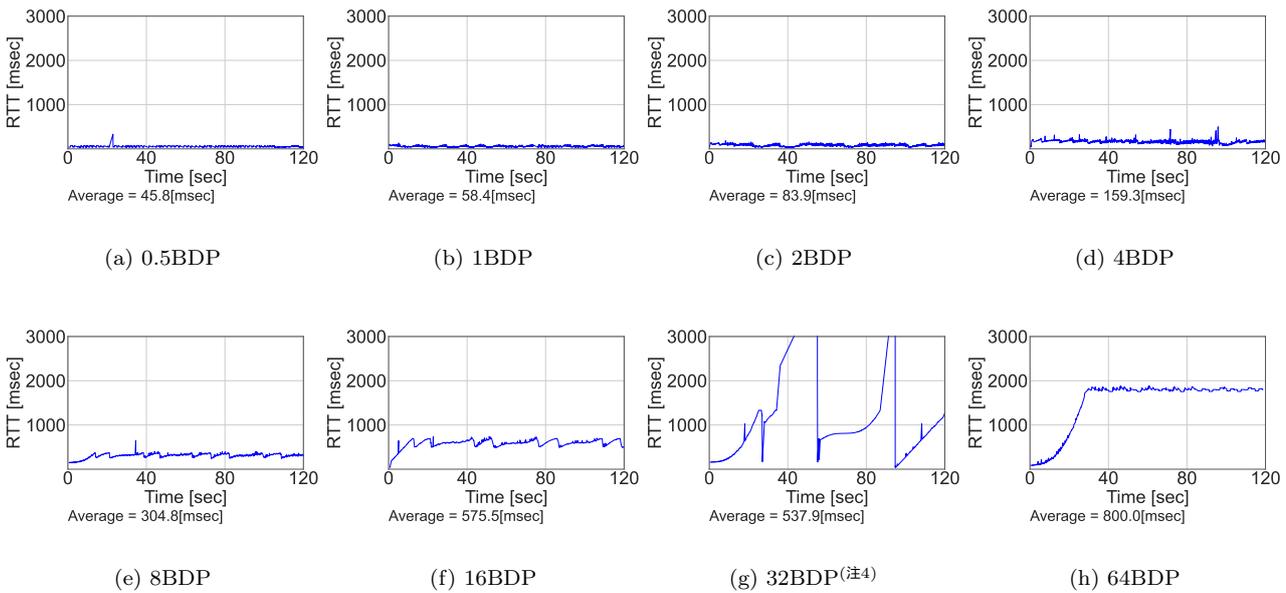


図 5: Copa フローの RTT の時間変化 (Copa フローと CUBIC フローが  $n = 1$  本ずつ競合)

競合するとき、スループットが不公平になる問題や、バッファサイズが大きくなるにしたがって、RTT が増加する現象が確認された。

## 6. Copa vs BBR

本章では、送信端末  $S_{1,1} \sim S_{1,n}$  における TCP 輻輳制御

(注3) Copa フローの送信中パケット数が増加しているのは、パケットロス率が大きくなり、累積確認応答が返ってきていないパケットを送信中パケットとして数えるためである。

(注4) RTT のピーク値は、時刻 55.09 [sec] における 4.12 [sec] と、時刻 94.74 [sec] における 4.39 [sec] である。このピーク値の RTT が非常に大きい原因は、再送タイムアウトによる。

機構が Copa、送信端末  $S_{2,1} \sim S_{2,n}$  における TCP 輻輳制御機構が BBR のときの性能評価の結果について述べる。

図 6(a) に Copa フローと BBR フローが  $n = 1$  本ずつ競合したとき、図 6(b) に  $n = 4$  本ずつ競合したときのそれぞれについて、バッファサイズに対する 1 フローあたりの平均スループットの変化を示す。図 7 に Copa フローと BBR フローが  $n = 1$  本ずつ競合したときにおける、Copa フローの送信中パケット数の時間変化、図 8 に BBR フローの送信中パケット数の時間変化を示す。図 9 に Copa フローと BBR フローが  $n = 1$  本ずつ競合したときにおける、Copa

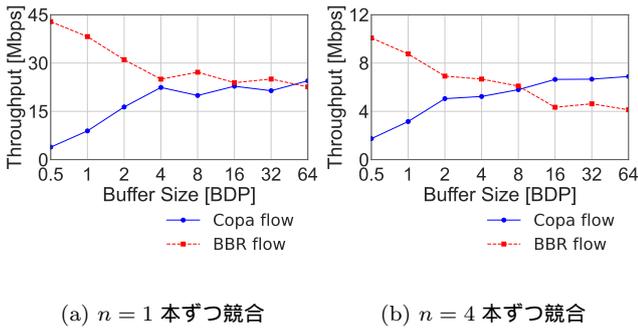


図 6: バッファサイズに対する 1 フローあたりの平均スループットの変化 (Copa vs BBR)

フローの RTT の時間変化を示す。図 10 に Copa フローと BBR フローが  $n = 4$  本ずつ競合したときにおける、Copa フロー 1 の送信中パケット数の時間変化、図 11 に BBR フロー 1 の送信中パケット数の時間変化を示す。図 12 に Copa フローと BBR フローが  $n = 4$  本ずつ競合したときにおける、Copa フロー 1 の RTT の時間変化を示す。

図 6(a) より、 $n = 1$  本ずつ競合したとき、バッファサイズが 2 [BDP] 以下の場合、Copa フローよりも BBR フローのほうがスループットが高く、バッファサイズが 4 [BDP] 以上の場合、Copa フローと BBR フローは比較的公平なスループットとなっていることがわかる。

図 7, 8 より、バッファサイズが 2 [BDP] 以下の場合、BBR フローのほうがスループットが高くなることは、Copa フローよりも BBR フローのほうが送信中パケット数が多いことに起因していることがわかる。また、バッファサイズが 2 [BDP] 以下の場合、図 9 より、RTT が大きくなっていないことがわかる。このとき、バッファリング遅延が小さいことから、Copa フローはバッファが空になることを検出するため、デフォルトモードで動作する割合が高くなっている。Copa フローのデフォルトモードでは、利用可能帯域を捕捉しようとする一方、BBR フローは、ProbeBW モードにおける送信レート  $\widehat{BtlBw}_T$  が利用可能帯域よりも大きくなる [15]。そのため、Copa フローよりも BBR フローのほうが送信中パケット数が多くなる。結果として、BBR フローのほうが Copa フローよりもスループットが高くなる。

図 9 より、バッファサイズが 4 [BDP] 以上の場合、RTT が 100 [msec] 以上となり、RTT が大きくなっているのがわかり、Copa フローは、競合モードで動作している割合が増えてくる。そのため、Copa フローのスループットが大きくなり、結果として、BBR フローと同程度のスループットとなっている。

また、図 6(b) より、 $n = 4$  本ずつ競合したとき、 $n = 1$  本ずつ競合したときと比べて、バッファサイズが 16 [BDP] 以上の場合にスループットが不公平になっていることがわ

かる。

図 10, 11 より、バッファサイズが 2 [BDP] 以下の場合、Copa よりも BBR フローの送信中パケット数が多いことがわかる。このとき、デフォルトモードで動作する割合の高い Copa フローと BBR フローが競合するため、BBR フローのほうが Copa フローよりもスループットが高くなっている。

図 12 より、バッファサイズが大きくなるにしたがい、RTT が大きくなっていることがわかり、Copa が競合モードで動作する割合も増加する。Copa フローが競合モードにあるとき、AIMD のロススペース輻輳制御機構に近い挙動をするため、Copa フローと ProbeBW モードの BBR フローが同程度のスループットとなる [8]。BBR フローは、おおよそ  $W_{R, BBR}$  おきに、ProbeRTT モードに入るので、その分、バッファサイズが大きい場合に、Copa フローよりもスループットで劣る。

このように、バッファリング遅延を抑える輻輳制御機構同士である Copa と BBR が競合するときにおいても、スループットが不公平になる問題や、バッファサイズが大きい場合に RTT が増加する現象が観測された。

## 7. まとめ

本稿では、異種の輻輳制御機構が競合するときの性能評価を行った。バッファリング遅延を抑える最新の輻輳制御機構である Copa フローと最大のシェアを持つロススペース輻輳制御機構である CUBIC フロー、及び Copa フローとバッファリング遅延を抑える輻輳制御機構としてシェアを伸ばしている BBR フローが競合したときにおいて、スループットが不公平になる問題が確認された。また、バッファリング遅延を抑える輻輳制御機構のフロー同士である Copa フローと BBR フローが競合したときにおいても、バッファサイズが大きい場合に RTT が増加する現象が観測された。

今後の課題として、Copa フローと CUBIC フローが競合するとき、及び Copa フローと BBR フローが競合するときにおけるスループット公平性に関する問題、さらに、バッファリング遅延を抑える輻輳制御機構のフロー同士である Copa フローと BBR フローが競合するときにおける RTT が増加する問題について、改善策を検討することが挙げられる。

謝辞 本研究の一部は、東北大学電気通信研究所共同プロジェクト研究 (H31/A24) と福島大学学内競争的研究資金 (21RG002) より実施されたものである。

## 参考文献

- [1] Jim: Gettys. "Bufferbloat: Dark buffers in the internet". *IEEE Internet Computing*, 15(3):96-96, (2011).
- [2] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, So-

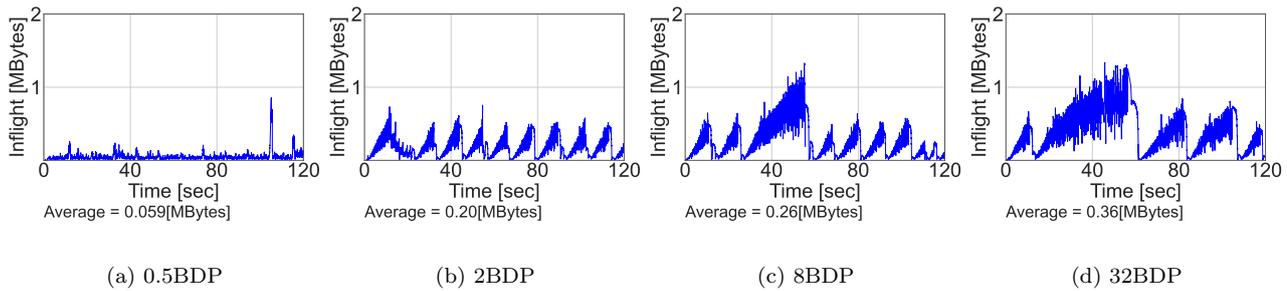


図 7: Copa フローの送信中パケット数の時間変化 (Copa フローと BBR フローが  $n = 1$  本ずつ競合)

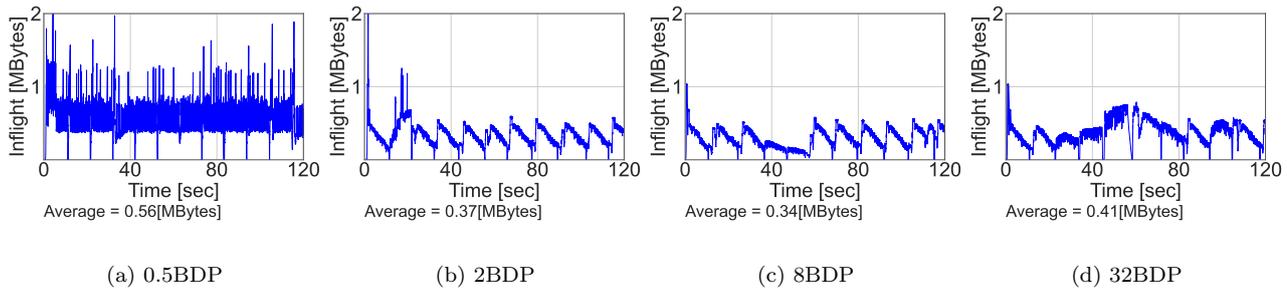


図 8: BBR フローの送信中パケット数の時間変化 (Copa フローと BBR フローが  $n = 1$  本ずつ競合)

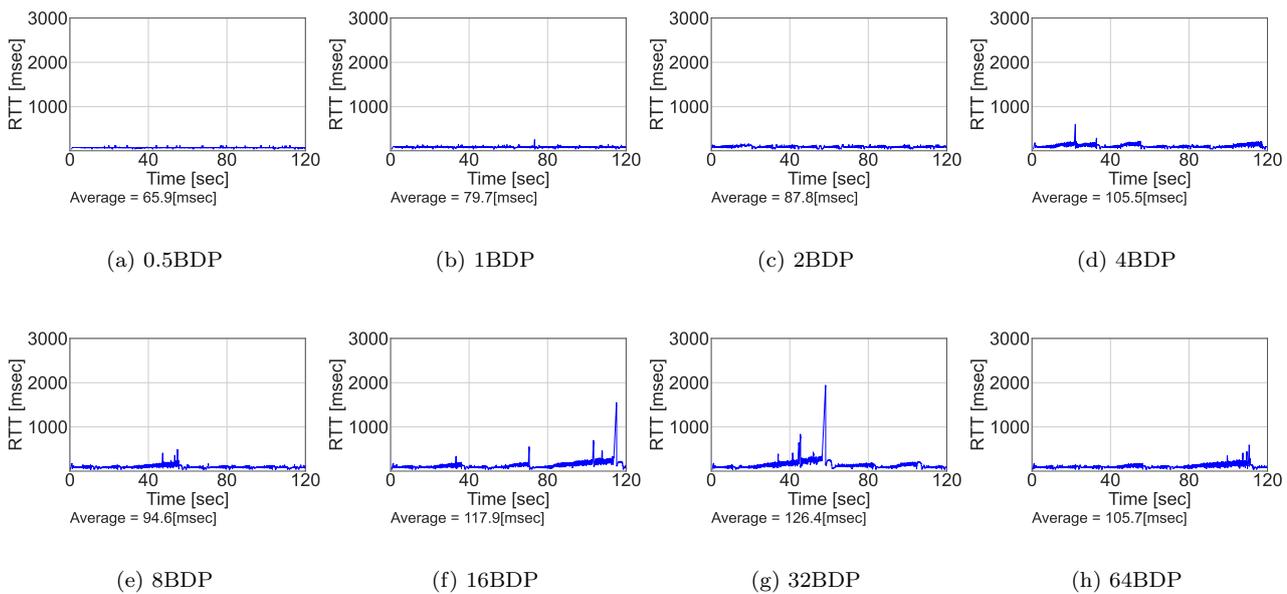


図 9: Copa フローの RTT の時間変化 (Copa フローと BBR フローが  $n = 1$  本ずつ競合)

heil Hassas Yeganeh, and Van: Jacobson. "BBR: Congestion-Based Congestion Control: Measuring bottleneck bandwidth and round-trip propagation time". *Queue*, 14(5):20–53, (2016).

[3] Nitin Garg:.. Copa congestion control for video performance - Facebook Engineering. 入手先 <<https://engineering.fb.com/2019/11/17/video-engineering/copa/>>. (2021.06.02).

[4] Jana Iyengar and Martin Thomson:.. "QUIC: A UDP-Based Multiplexed and Secure Transport". RFC 9000, (2021).

[5] Vankat Arun and Hari Barakrishnan:.. "copa: Practical delay-based congestion control for the internet". 入手先 <<https://web.mit.edu/copa/#facebook>>. (2021.06.17).

[6] Sangtae Ha, Injong Rhee, and Lisong: Xu. "CUBIC: a new TCP-friendly high-speed TCP variant". *ACM SIGOPS operating systems review*, 42(5):64–74, (2008).

[7] Ayush Mishra, Xiangpeng Sun, Atishya Jain, Sameer Pande, Raj Joshi, and Ben: Leong. "The great internet TCP congestion control census". *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 3(3):1–24, (2019).

[8] Ranysha Ware, Matthew K Mukerjee, Srinivasan Seshan, and Justine: Sherry. "Modeling BBR's interactions with loss-based congestion control". In *Proceedings of the Internet Measurement Conference*, pages 137–143, (2019).

[9] Dominik Scholz, Benedikt Jaeger, Lukas Schwaighofer,

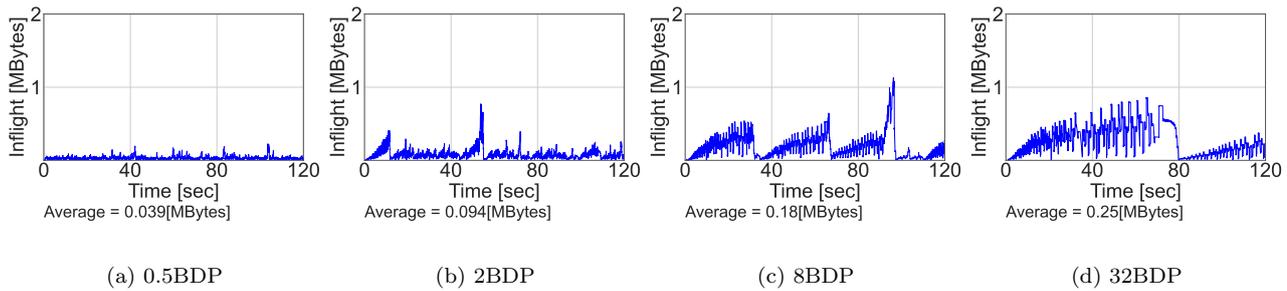


図 10: Copa フロー 1 の送信中パケット数の時間変化 (Copa フローと BBR フローが  $n = 4$  本ずつ競合)

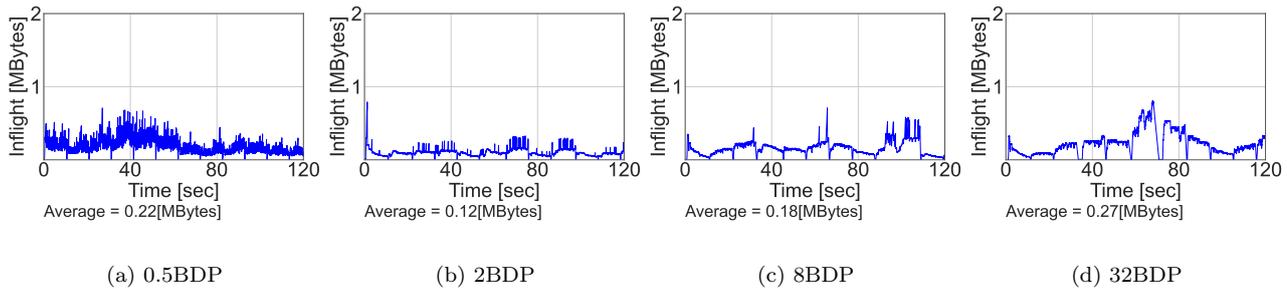


図 11: BBR フロー 1 の送信中パケット数の時間変化 (Copa フローと BBR フローが  $n = 4$  本ずつ競合)

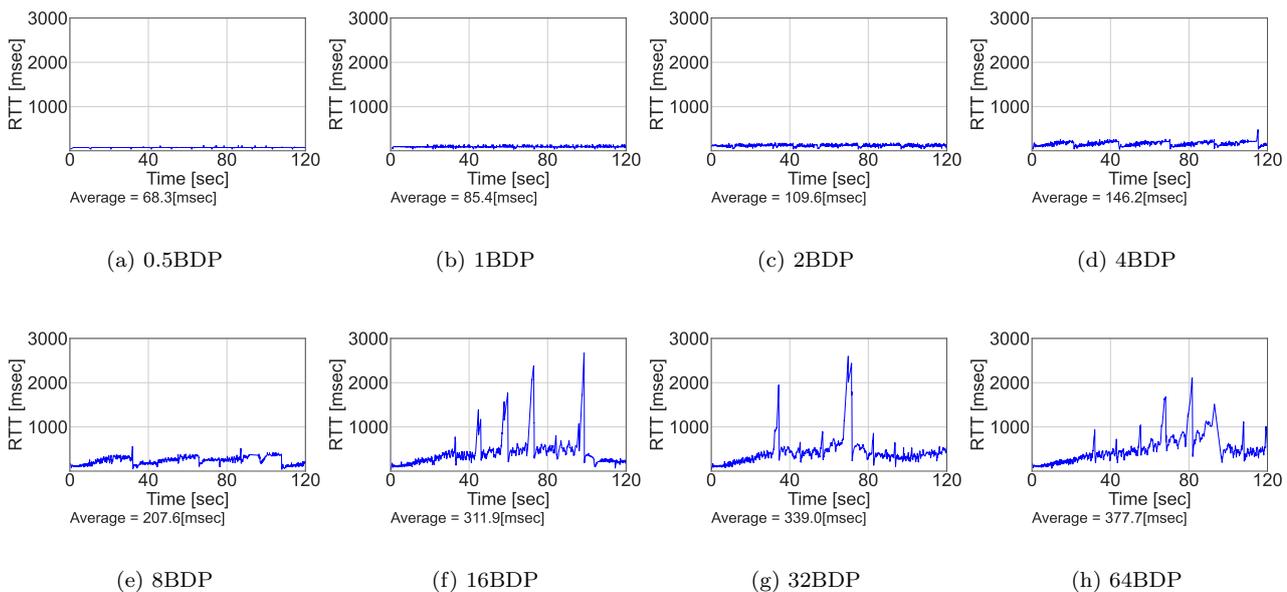


図 12: Copa フロー 1 の RTT の時間変化 (Copa フローと BBR フローが  $n = 4$  本ずつ競合)

- Daniel Raumer, Fabien Geyer, and Georg Carle. "Towards a deeper understanding of TCP BBR congestion control". In *2018 IFIP networking conference (IFIP networking) and workshops*, pages 1–9. IEEE, (2018).
- [10] "Mininet: An Instant Virtual Network on Your Laptop (or Other PC) - Mininet.". 入手先 <<http://mininet.org/>>. (2021.04.14).
- [11] Mo Dong, Qingxi Li, Doron Zarchy, P Brighten Godfrey, and Michael Schapira. {PCC}: Re-architecting congestion control for consistent high performance. In *12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)*, pages 395–408, 2015.
- [12] "Introduction - CCP Guide". 入手先 <<https://ccp-project.github.io/ccp-guide/intro.html>>. (2021.06.02).
- [13] The Tcpdump Group. Tcpdump/libpcap public repository. 入手先 <<https://www.tcpdump.org/>>. (2021.06.07).
- [14] Shawn Ostermann. tcptrace(1) - linux man page. 入手先 <<https://linux.die.net/man/1/tcptrace>>. (2021.06.07).
- [15] Mario Hock, Roland Bless, and Martina Zitterbart. "Experimental evaluation of BBR congestion control". In *2017 IEEE 25th International Conference on Network Protocols (ICNP)*, pages 1–10. IEEE, (2017).