HOKKAIDO UNIVERSITY

Title	環境の変化に適応可能な強化学習利用型輻輳制御の基礎的評価
Author(s)	青木, 一真; 三橋, 力麻; 飯田, 勝吉 他
Description	電子情報通信学会 情報ネットワーク研究会 (IN)/ネットワークシステム研究会 (NS). 令和7年3月6日～令和7年3月7日. 沖縄産業支援センター、沖縄県. IEICE Technical Committee on Information Networks (IN)/Technical Committee on Network Systems (NS). 2025/03/06-2025/03/07. Okinawa Industry Support Center, Okinawa.
Citation	電子情報通信学会技術研究報告, 124(419), 274-280
Issue Date	2025-02-27
Doc URL	https://hdl.handle.net/2115/94120
Rights	Copyright ©2025 IEICE
Type	journal article
File Information	NS2024-242.pdf

Hokkaido University Collection of Scholarly and Academic Papers : HUSCAP

一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
NS2024-242(2025-03)

環境の変化に適応可能な強化学習利用型輻輳制御の基礎的評価

青木　一真†　三橋　力麻††　飯田　勝吉†††　高井　昌彰†††

† 北海道大学 大学院情報科学院 情報理工学コース 先端ネットワーク研究室
〒060–0811 札幌市北区北 11 条西 5 丁目
†† , ††† 北海道大学 情報基盤センター 〒060–0811 札幌市北区北 11 条西 5 丁目
E-mail: †aoki.kazuma.s1@elms.hokudai.ac.jp, ††mitsuhashi@tanega.iic.hokudai.ac.jp,
†††{iida,ytakai}@iic.hokudai.ac.jp

あらまし　インターネットが社会基盤として不可欠な存在となる中、ネットワークの輻輳制御は通信品質を維持する上で重要な役割を担っている。NewReno や CUBIC をはじめとする多くの輻輳制御アルゴリズムは、事前に定めたルールに基づく静的なアルゴリズムとして動作するため、ネットワーク環境の変化に応じた最適化が難しい問題がある。この問題を解決するため、強化学習を利用した動的なアルゴリズムによる輻輳制御の研究が始まっている。しかしながら、先行研究では行動の選択や報酬の与え方といった強化学習としての基本的な評価が十分に行われていない。そこで本研究では、深層強化学習を用いた新しい輻輳制御アルゴリズム「RL-NewReno」を実装し、その性能評価を行う。実験の結果、提案手法は既存の NewReno に比べて平均スループットが最大で約 2% 向上することを確認した。
キーワード　輻輳制御，機械学習，強化学習.

Preliminary evaluation of a congestion control method using reinforcement learning for adopting environment changes

Kazuma AOKI†, Rikima MITSUHASHI††, Katsuyoshi IIDA†††, and Yoshiaki TAKAI†††

† Laboratory of Advanced Information Network,
Course of Computer Science and Information Technology, Graduate School of Information Science & Engineering,
Hokkaido University, Kita 11 Nishi 5, Kita-ku, Sapporo, 060–0811, Japan.
†† , ††† Information Initiative Center, Hokkaido University, Kita 11 Nishi 5, Kita-ku, Sapporo, 060–0811, Japan.
E-mail: †aoki.kazuma.s1@elms.hokudai.ac.jp, ††mitsuhashi@tanega.iic.hokudai.ac.jp,
†††{iida,ytakai}@iic.hokudai.ac.jp

Abstract　As the Internet becomes an indispensable part of our social infrastructure, congestion control plays an important role in maintaining communication quality. Many congestion control algorithms, such as NewReno and CUBIC, operate as static algorithms based on predefined rules, and there is a problem with the difficulty of optimizing them in response to changes in the network environment. To solve this problem, research efforts have begun on congestion control using dynamic algorithms that utilize reinforcement learning. However, in the existing researches, the basic evaluation of reinforcement learning, such as the selection of actions and the way in which rewards are given, has not been sufficiently carried out. Therefore, in this research, we proposed and implemented the new congestion control algorithm *RL-NewReno* using deep reinforcement learning and evaluated its performance. Through simulation evaluations, we confirmed that the proposed method improved the average throughput by about 2% at maximum compared to the existing NewReno.
Key words　Congestion control, machine learning, reinforcement learning.

1. はじめに

インターネットが社会基盤として不可欠な存在となる中、ネットワークの輻輳制御は通信品質を維持する上で重要な役割を担っている。輻輳制御は、トランスポート層の TCP プロトコルに組み込まれた機能の一つであり、回線の利用が集中して混雑している状態を検知し、データ送信量を調整する。データ送信量の調整は輻輳ウィンドウ（cwnd）の大きさを変化させること

This article is a technical report without peer review, and its polished and/or extended version may be published elsewhere.
Copyright ©2025 by IEICE

によって通信量をコントロールしており、その動作は輻輳制御アルゴリズムによって規定される。代表的な輻輳制御アルゴリズムは、NewReno や CUBIC などパケットロスを観測するロスベース方式、Vegas や BBR などパケットの RTT を観測する遅延ベース方式、Illinois や YeAH など両方の方式を使い分けるハイブリッド方式に分類できる。これらの方式はいずれも事前に定めたルールに基づく静的なアルゴリズムとして動作するため、ネットワーク環境の変化に応じた最適化が難しい問題がある。

この静的な輻輳制御アルゴリズムの問題を解決するため、強化学習を利用した動的なアルゴリズムによる輻輳制御の研究が始まっている [1], [2]。しかしながら、先行研究では通信品質の向上に一定の効果のあるアルゴリズムを提案しているものの、行動の選択や報酬の与え方といった深層強化学習としての基本的な評価が十分に行われていない。

そこで本研究では、深層強化学習を用いた新しい輻輳制御アルゴリズム「RL-NewReno」を実装し、その性能評価を行う。「RL-NewReno」は NewReno をベースに深層強化学習を組み込んだ輻輳制御アルゴリズムである。輻輳制御における cwnd の増加方法として 4 種類の行動空間を 2 パターン用意した。性能評価は ns3 を用いたシミュレーション環境において 128Mbps の回線と 512Mbps の回線を用意し、RL-NewReno の単一フローが流れている状態と NewReno と RL-NewReno の 2 フローが流れている状態を評価した。実験の結果、提案手法は既存の NewReno に比べて平均スループットが最大で約 2% 向上することを確認した。

本研究の貢献は以下のとおりである。

- 深層強化学習による新しい輻輳制御手法を提案した。
- 輻輳制御に適した深層強化学習の行動空間を評価した。
- 提案手法が NewReno の性能を上回ることを示した。

2. 関 連 研 究

機械学習による動的な輻輳制御アルゴリズムの研究開発が始まっている [1]。最初の手法 Remy [3] は、人間がアルゴリズムを考案・調整するアプローチ (NewReno など) をとらず、シミュレーションを用いたオフライン学習により適切なアルゴリズムを機械的に導出する。この方式には、想定していないネットワーク環境に変化した場合に対応ができない課題がある。

その後、強化学習を使った輻輳制御手法 QTCP [2] が提案されている。QTCP は強化学習に基づくため、時々刻々とネットワーク環境が変化する状況において、直前に獲得した統計データに基づくオンラインでの輻輳制御の最適化が可能となる。しかし、QTCP は単純な強化学習だけを用いるのではなく、Kenarva コーディングという状態削減手法を併用しているため、強化学習単体で構成した輻輳制御アルゴリズムの性能を明らかにしていない。さらに、強化学習の各ステップにおける行動の選択肢として、CWND の増減（10Byte 増、1Byte 減、増減なし）の 3 つの行動が設定されているのみで、行動の選択肢の最適化がなされていない。

また、文献 [4] と文献 [5] において、ネットワークシミュレータ ns3 [6] と機械学習ライブラリを連携させたシミュレーションを可能とするフレームワーク (Open AI gym, ns3-ai) が提案されている。いずれのフレームワークにも、強化学習を用いた TCP 輻輳制御がサンプルプログラムとして含まれている（サンプルプログラム名=RL-TCP）。文献 [4] によると QTCP が参考文献としてあげられているが、QTCP の強化学習手法とは異なる強化学習手法が用いられている。具体的には、RL-TCP には Kanerva コーディングによる状態削減手法は用いられておらず、また、行動の選択肢や報酬の与え方も異なっている。文献 [4], [5] の RL-TCP は、ネットワークシミュレータ ns3 と機械学習ライブラリが連携できることを示すための実例として提示されているのみで、輻輳制御のための強化学習の最適化が行われてはいない。

表 1　RL-NewReno-SS の行動空間

行動 1	式 1（スロースタート）
行動 2	式 2, $\alpha = 1$
行動 3	式 2, $\alpha = 2$
行動 4	式 2, $\alpha = 3$

表 2　RL-NewReno-noSS の行動空間

行動 1	式 2, $\alpha = 4$
行動 2	式 2, $\alpha = 1$
行動 3	式 2, $\alpha = 2$
行動 4	式 2, $\alpha = 3$

3. RL-NewReno アルゴリズムの提案

前節で記載した通り既存の強化学習を用いる輻輳制御方式は、強化学習を用いた輻輳制御の標準方式とはなりえていない。そこで、本研究では強化学習を用いた輻輳制御の標準方式の構築を目指す。

具体的には NewReno をベースアルゴリズムとし、深層強化学習を組みこむことで RL-NewReno を構成する。RL-NewReno は NewReno の cwnd 増加の動作アルゴリズムを変更し、2 種類のバージョンを用意する。1 種類目は行動空間にスロースタートを含めた RL-NewReno-SS で、表 1 に示す。2 種類目は行動空間にスロースタートを含めない RL-NewReno-noSS で、表 2 に示す。なお、RL-NewReno-SS の式 1 はスロースタートと呼ばれ、通信開始時などに指数的に輻輳ウィンドウサイズ (cwnd) を増加させる式である。RL-NewReno-SS および RL-NewReno-noSS には、行動空間にそれぞれ 4 つの行動を用意し、cwnd の増加の仕方を α で変更する。また、パケットロス検出時の cwnd の減少の仕組みは、NewReno と同じアルゴリズムを用いる。

$$\text{new_cwnd} = \text{cwnd} + \text{segmentSize} \tag{1}$$

$$\text{new_cwnd} = \text{cwnd} + \alpha \times \max\left(1, (\text{segmentSize})^2/\text{cwnd}\right) \tag{2}$$

また、深層強化学習のモデルと報酬については、ns3-ai [5] と同じものを採用した（モデル＝表 3、報酬＝式 3)。

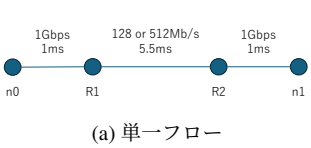
$$\text{reward} = \text{segmentsAcked} - \text{bytesInFlight} - \text{cwnd} \tag{3}$$

4. 性 能 評 価

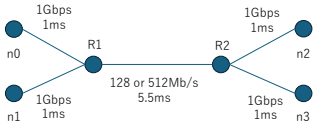
本節では提案手法 RL-NewReno の機能性能を明らかにするためコンピュータシミュレーションにより性能を評価する。具体

表 3　RL-NewReno のニューラルネットワークモデル

入力層	Linear (5 ノード)
中間層	Linear (20 ノード)
出力層	Linear (4 ノード)
損失関数	MSE
最適化関数	Adam
学習率	0.0001



(a) 単一フロー



(b) 2 フロー

図 1　シミュレーション環境

的には、提案手法をネットワークシミュレータ ns3 [6] 上に実装してシミュレーション実験を行なう。また、深層強化学習については、ns3 から ns3-ai [5] の API を通じて PyTorch のライブラリを呼び出すことで実装する。

4.1　シミュレーション環境

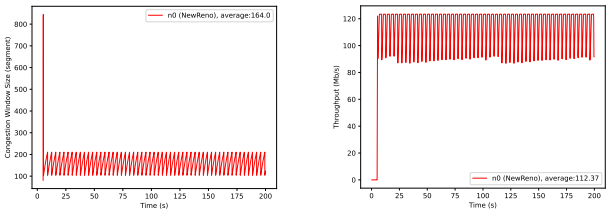
シミュレーション環境を図 1 に示す。単一フロー時はクライアント（n0）がサーバ（n1）と通信する。その間には中継ルータを 2 つ用意し (R1, R2)、クライアントとルータ、ルータとサーバ間にアクセスリンクを作成する。また、ルータ間にボトルネックリンクを作成する。2 フロー時（競合フローが存在する場合）にはダンベルトポロジでシミュレーションを行い、n0 は n2 と、n1 は n3 と通信を行う。このとき、表 4 に記載の 3 つのシナリオでシミュレーション実験を行う。なお、n0 と n1 が近いタイミングで通信を開始するシナリオ 1 とシナリオ 2 では、輻輳制御の動作を安定させるため、n0 の通信開始後から 10 秒後に n1 が通信を開始する。また、実験に用いた計算機とソフトウェアの環境を表 5 に示す。

表 4　2 フロー時（競合フロー存在環境）の実験シナリオ

シナリオ 1	n0, n1 が通信を開始し、同時に終了する
シナリオ 2	n0, n1 が通信を開始し、n0 が先に通信を終了する
シナリオ 3	n0 が先に通信を開始し、n1 が後から通信を開始する

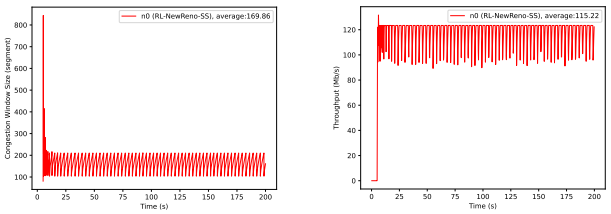
表 5　実 験 環 境

種別	諸元 / バージョン
GPU	NVidia GeForce RTX3080 / RTX3090
OS	Ubuntu 22.04.4 LTS
ns-3	3.38
ns3-ai	1.2.0
python	3.10.12
Pytorch	libtorch-cxx11



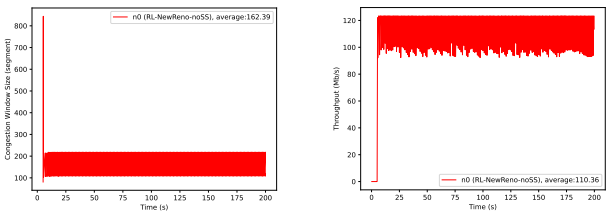
(a) cwnd　(b) throughput

図 2　単一フロー, NewReno, BW=128Mb/s の結果



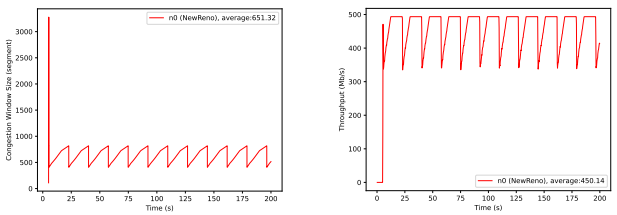
(a) cwnd　(b) throughput

図 3　単一フロー, RL-NewReno-SS, BW=128Mb/s の結果



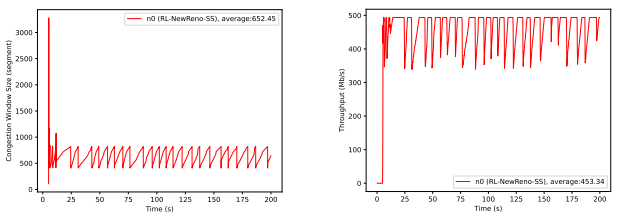
(a) cwnd　(b) throughput

図 4　単一フロー, RL-NewReno-noSS, BW=128Mb/s の結果



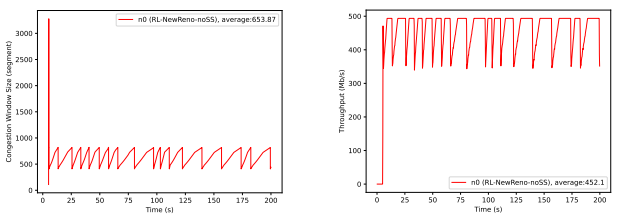
(a) cwnd　(b) throughput

図 5　単一フロー, NewReno, BW=512Mb/s の結果



(a) cwnd　(b) throughput

図 6　単一フロー, RL-NewReno-SS, BW=512Mb/s の結果



(a) cwnd　(b) throughput

図 7　単一フロー, RL-NewReno-noSS, BW=512Mb/s の結果

4.2 単一フローの実験結果

単一フローの結果を示す。

4.2.1 ボトルネックリンク帯域が 128Mb/s の場合

ボトルネックリンク帯域が 128Mb/s の場合のシミュレーション結果（cwnd, throughput）について、NewReno、RL-NewReno-SS、RL-NewReno-noSS のそれぞれを図 2 から図 4 に示す。

図 2 (b) と図 3 (b) を比較すると、NewReno のスループットの平均値は 112.37Mb/s であり RL-NewReno-SS のスループットの平均値は 115.22Mb/s であったことから、RL-NewReno-SS は NewReno の性能を 2.54% 上回った。一方で、図 2 (b) と図 4 (b) を比較すると、NewReno のスループットの平均値は 112.37Mb/s であり RL-NewReno-noSS のスループット平均値は 110.36Mbps であったことから、RL-NewReno-noSS は NewReno の性能を下回った。

4.2.2 ボトルネックリンク帯域が 512Mb/s の場合

ボトルネックリンク帯域が 512Mb/s の場合のシミュレーション結果（cwnd, throughput）について、NewReno、RL-NewReno-SS、RL-NewReno-noSS のそれぞれを図 5 から図 7 に示す。

図 5 (b) と図 6 (b) を比較すると、NewReno のスループットの平均値は 450.14Mb/s であり RL-NewReno-SS のスループットの平均値は 453.34Mb/s であったことから、RL-NewReno-SS は NewReno の性能を 0.71% 上回った。また、図 5 (b) と図 7 (b) を比較すると、NewReno のスループットの平均値は 450.14Mb/s であり RL-NewReno-noSS のスループットの平均値は 452.1M/ps であったことから、RL-NewReno-noSS は NewReno の性能を 0.43% 上回った。

以上の結果を踏まえると、RL-NewReno-SS は NewReno のスループットを最大 2.54% 上回っており、効率的に帯域を利用できていることが確認できた。また、行動空間については、RL-NewReno-SS の方が RL-NewReno-noSS よりも性能向上に寄与しているといえる。

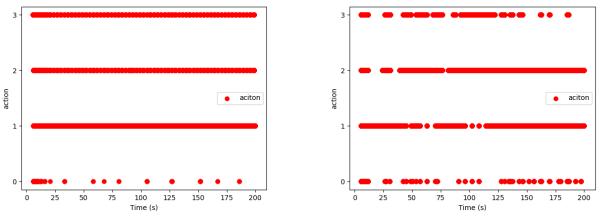
4.2.3 行動選択に関する考察

RL-NewReno-SS の行動選択の推移グラフを図 8 に示す。

図 8(a) によると、選択された行動が、行動 1、行動 2、行動 3 であることが多く、行動 0 の選択回数が少ない。RL-NewReno-SS がボトルネックリンク帯域が 128Mb/s において最も平均スループットが高かったことを踏まえると、最適な行動選択には行動 0(スロースタート) がほとんど必要ない可能性が高い。

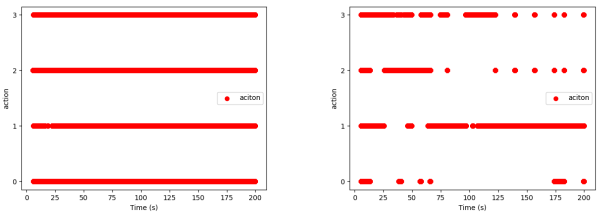
また、図 8(a) と図 8(b) を比較すると、ボトルネックリンク帯域が 512Mb/s の時、行動 0 (スロースタート) を選択する回数が増えている。RL-NewReno-SS がボトルネックリンク帯域が 512Mb/s において最も平均スループットが高かったことを踏まえると、ボトルネックリンク帯域が 512Mb/s においては最もスループットを高める行動 0 を選択する回数を増やし、広帯域を埋めるような学習がなされたと考えられる。

次に、RL-NewReno-noSS の行動選択の推移グラフを図 9 に示す。図 9(a) のボトルネック帯域が 128Mb/s の状況においては、4 種の行動がほぼ同数バランスよく選択されている、一方 9(b) の 512Mb/s の状況においては、行動 0 がほとんど選択されておらず、また、行動 1 が選択される回数が行動 2, 3 と比べて



(a) BW=128Mb/s　　(b) BW=512Mb/s

図 8　単一フロー, RL-NewReno-SS の行動履歴



(a) BW=128Mb/s　　(b) BW=512Mb/s

図 9　単一フロー, RL-NewReno-noSS の行動履歴

多いことがわかる。これは、異なるボトルネック帯域幅が与えられたとき、選択すべき行動の傾向が変わるためと考えられる。

4.3 競合フローが存在する場合（2 フロー）

次に、競合フローが存在する場合 (2 フロー) の結果を示す。4.1 節で述べた通り、シナリオ 1（2 フローが同時に通信開始し、同時に終了する)、シナリオ 2（2 フローが同時に通信開始し、n0 フロー（競合フロー）が先に通信を終了する)、シナリオ 3（n0 フロー（競合フロー）が先に通信し、後から n1 フロー（対象フロー）が合流する）の 3 つのシナリオで評価する (表 4 参照)。

4.3.1 シナリオ 1

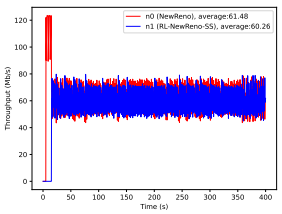
シナリオ 1 は、NewReno フローと RL-NewReno のフローがほぼ同時に通信を開始し、同時に終了するシナリオである。スループットのシミュレーション結果を図 10, 11 に示す。

a)　ボトルネックリンク帯域が 128Mb/s の時の結果

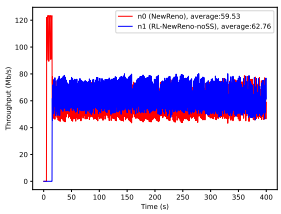
図 10(a) によると、RL-NewReno-SS, NewReno の共存シナリオにおいて、それぞれのフローの平均スループットは 60.26, 61.48Mb/s であり、公平性を評価する Jain's Fairness Index は 0.9999 であった。図 10(b) によると、RL-NewReno-noSS, NewReno の共存シナリオにおいて、それぞれのフローの平均スループットは 62.76, 59.33Mb/s であり、Jain's Fairness Index は 0.9993 であった。このことから、RL-NewReno-SS, RL-NewReno-noSS はともに NewReno との公平性が実現できていることがわかる。

b)　ボトルネックリンク帯域が 512Mb/s の時の結果

図 11(a) によると、RL-NewReno-SS, NewReno の共存シナリオにおいて、それぞれのフローの平均スループットは 246.60, 239.48Mb/s であり、Jain's Fairness Index は 0.9998 であった。図 11(b) によると、RL-NewReno-noSS, NewReno の共存シナリオにおいて、それぞれのフローの平均スループットは 244.30, 241.2Mb/s であり、Jain's Fairness Index は 0.9999 であった。

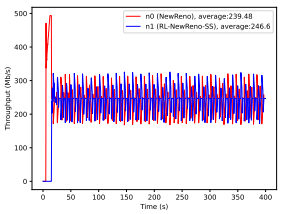


(a) NewReno & RL-NewReno-SS

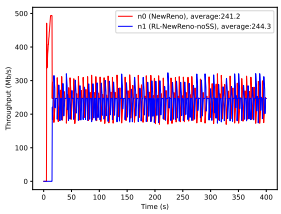


(b) NewReno & RL-NewReno-noSS

図 10　2 フロー, BW=128Mb/s, シナリオ 1 のスループット

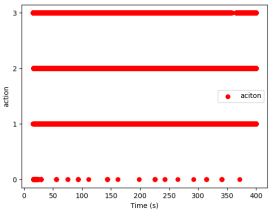


(a) NewReno & RL-NewReno-SS

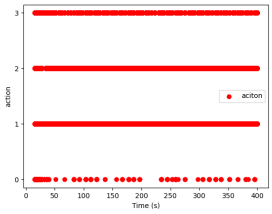


(b) NewReno & RL-NewReno-noSS

図 11　2 フロー, BW=512Mb/s, シナリオ 1 のスループット

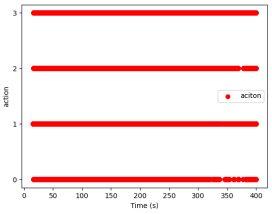


(a) BW = 128Mb/s

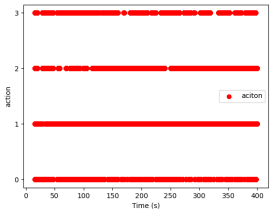


(b) BW = 512Mb/s

図 12　2 フロー, RL-NewReno-SS, シナリオ 1 の行動履歴

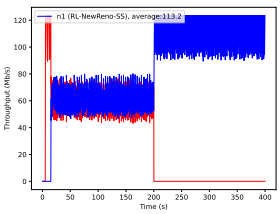


(a) BW=128Mb/s

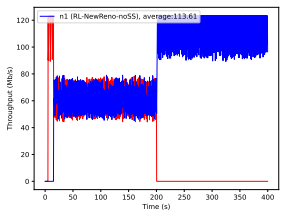


(b) BW=512Mb/s

図 13　2 フロー, RL-NewReno-noSS, シナリオ 1 の行動履歴

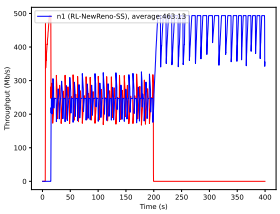


(a) NewReno & RL-NewReno-SS

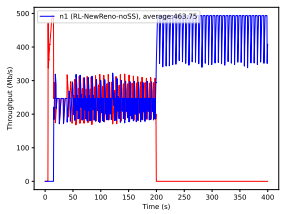


(b) NewReno & RL-NewReno-noSS

図 14　2 フロー, BW=128Mb/s, シナリオ 2 のスループット

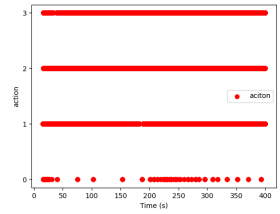


(a) NewReno & RL-NewReno-SS

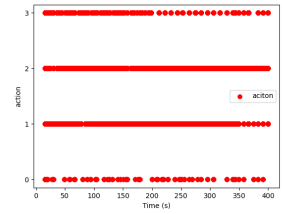


(b) NewReno & RL-NewReno-noSS

図 15　2 フロー, BW=512Mb/s, シナリオ 2 のスループット

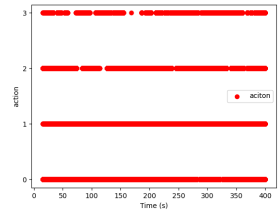


(a) BW = 128Mb/s

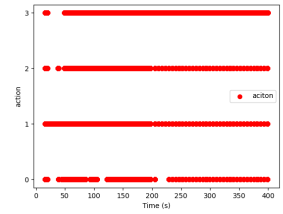


(b) BW = 512Mb/s

図 16　2 フロー, RL-NewReno-SS, シナリオ 2 の行動履歴



(a) BW = 128Mb/s



(b) BW = 512Mb/s

図 17　2 フロー, RL-NewReno-noSS, シナリオ 2 の行動履歴

このことから、RL-NewReno-SS, RL-NewReno-noSS はともに NewReno との公平性が実現できていることがわかる。

c）　行動選択に関する考察

図 12, 13 に、RL-NewReno-SS, RL-NewReno-noSS の行動選択の履歴を示す。図 12 によると、ボトルネック帯域 128Mb/s, 512Mb/s を比較したとき、行動 0 (スロースタート) が選択される回数が全体的に増えており、RL-NewReno-SS には広帯域環境への適応能力があることが見て取れる。

一方で、図 13 によると、128Mb/s, 512Mb/s を比較したとき、行動の選択回数に大きな差はなく、RL-NewReno-noSS ではボトルネック帯域幅に応じた適応能力が低い可能性がある。

4.3.2　シナリオ 2

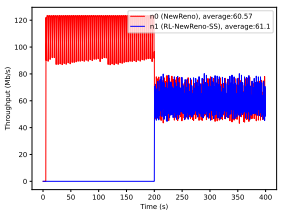
シナリオ 2 は、NewReno と RL-NewReno の 2 フローが同時に通信を開始し、先に NewReno が終了したのち、RL-NewReno だけが残るシナリオである。このシナリオのスループットのシミュレーション結果を図 14, 15 に示す。スループットの平均値は NewReno のフローのパケット送信が終了してからシミュレーション終了までの時間で計測している。

a）　ボトルネックリンク帯域が 128Mb/s の時の結果

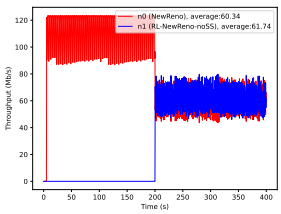
図 14 によると、NewReno フローの通信終了後の帯域利用率は、RL-NewReno-SS で約 88%、RL-NewReno-noSS で約 89% であった。このことから、両者ともに空き帯域の発生という環境変化に適応できていることが確認できる。

b）　ボトルネックリンク帯域が 512Mb/s の時の結果

図 15 によると、NewReno フローが通信を終了した後の帯域利用率は、RL-NewReno-SS で約 90%、RL-NewReno-noSS で約

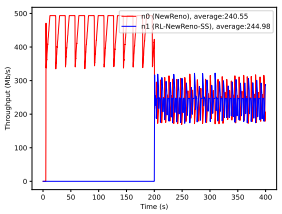


(a) NewReno & RL-NewReno-SS

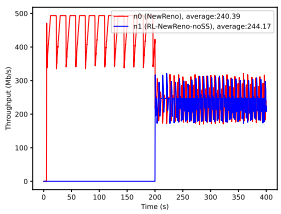


(a) NewReno & RL-NewReno-noSS

図 18　2 フロー, BW=128Mb/s, シナリオ 3 のスループット

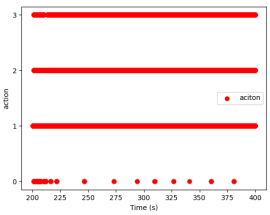


(a) NewReno & RL-NewReno-SS

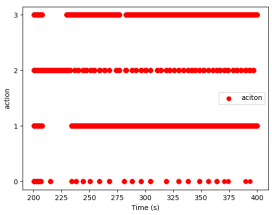


(a) NewReno & RL-NewReno-noSS

図 19　2 フロー, BW=512Mb/s, シナリオ 3 のスループット

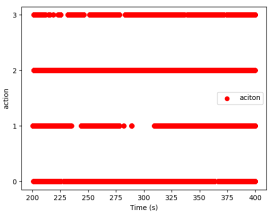


(a) BW = 128Mb/s

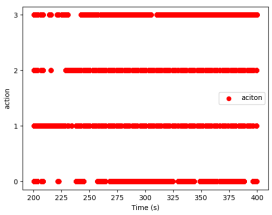


(b) BW = 512Mb/s

図 20　2 フロー, RL-NewReno-SS, シナリオ 3 の行動履歴



(a) BW = 128Mb/s



(b) BW = 512Mb/s

図 21　2 フロー, RL-NewReno-noSS, シナリオ 3 の行動履歴

91% であった。このことから、広帯域下でも空き帯域の発生という環境変化に適応出来ていることが分かる。

c）　行動選択に関する考察

図 16, 17 に、RL-NewReno-SS, RL-NewReno-noSS の行動選択の履歴を示す。図 16 において NewReno が通信を終了する 200 秒直後に着目すると、ボトルネック帯域 128Mb/s の時、行動 0 (スロースタート) の選択回数が増加している。また、ボトルネック帯域 512Mb/s の時、行動 3 の選択回数が減少している。つまり、RL-NewReno-SS では、環境変化が起こる 200 秒を境に、行動選択の傾向に変化が見られており、環境変化に適応する様子が見て取れる。

一方で、図 17 によると、ボトルネック帯域が 128Mb/s, 512Mb/s の両方の場合において、環境変化に合わせた行動選択を行っているような様子は見て取れない。

4.3.3　シナリオ 3

シナリオ 3 は、NewReno フローが先に通信を開始し、後から RL-NewReno のフローが合流するシナリオである。このシナリオのスループットのシミュレーション結果を図 18, 19 に示す。スループットの平均値は、RL-NewReno フローが通信を開始してからシミュレーション終了までの時間で計測している。

a）　ボトルネックリンク帯域が 128Mb/s の時の結果

図 18(a) によると、RL-NewReno-SS, NewReno の共存シナリオにおいて、それぞれのフローの平均スループットは 61.1, 60.57Mb/s であり、公平性を評価する Jain's Fairness Index は 0.9999 であった。図 18(b) によると、RL-NewReno-noSS, NewReno の共存シナリオにおいて、それぞれのフローの平均スループットは 61.74, 60.34Mb/s であり、Jain's Fairness Index は 0.9999 であった。このことから、RL-NewReno-SS, RL-NewReno-noSS はともに NewReno との公平性が実現できていることがわかる。

b）　ボトルネックリンク帯域が 512Mb/s の時の結果

図 19(a) によると、RL-NewReno-SS, NewReno の共存シナリオにおいて、それぞれのフローの平均スループットは 244.98, 240.55Mb/s であり、公平性を評価する Jain's Fairness Index は 0.9999 であった。図 19(b) によると、RL-NewReno-noSS, NewReno の共存シナリオにおいて、それぞれのフローの平均スループットは 244.17, 240.39Mb/s であり、Jain's Fairness Index は 0.9999 であった。このことから、RL-NewReno-SS, RL-NewReno-noSS はともに NewReno との公平性が実現できていることがわかる。

c）　行動選択に関する考察

図 20, 21 に、RL-NewReno-SS, RL-NewReno-noSS の行動選択の履歴を示す。図 20 によると、ボトルネック帯域 128Mb/s, 512Mb/s を比較したとき、行動 0 (スロースタート) が選択される回数が全体的に増えており、シナリオ 1 と同様に、RL-NewReno-SS には広帯域環境への適応能力があることが見て取れる。一方で、図 21 によると、128Mb/s, 512Mb/s を比較したとき、行動の選択回数に大きな差はなく、こちらもシナリオ 1 と同様に、RL-NewReno-noSS ではボトルネック帯域幅に応じた適応能力が低い可能性がある。

4.3.4　2 フロー時の評価のまとめ

シナリオ 1 からシナリオ 3 の結果を総括すると、シナリオ 1 およびシナリオ 3 では、RL-NewReno-SS と RL-NewReno-noSS の双方が高い公平性を実現できることが確認された。さらに、2 種類のボトルネックリンク帯域における深層強化学習の行動選択履歴を比較した結果、RL-NewReno-SS は広帯域環境においてスループットをより高める行動を選択する傾向があり、高い適応能力を持つ可能性があると考えられる。

一方、シナリオ 2 では、RL-NewReno-SS と RL-NewReno-noSS の両方が、帯域に空きが生じた際にその分を有効に活用できることが分かった。また、行動選択履歴の分析によると、RL-

NewReno-SS には環境変化に応じて行動選択の傾向が変化する様子が見られたが、RL-NewReno-noSS にはそれが見られなかった

5. おわりに

事前に定めたルールに基づく静的な輻輳アルゴリズムはネットワーク環境の変化に応じた最適化が難しい課題がある。この問題を解決するため、機械学習を活用した輻輳制御アルゴリズムの研究が進められており、特に強化学習を用いた手法は、リアルタイムで環境に適応できる点で有望視されている。しかしながら、先行研究では行動の選択や報酬の与え方といった強化学習としての基本的な評価が十分に行われておらず、その実用性や最適化に向けては未解明な点が多い。

本研究では、深層強化学習を用いた輻輳制御アルゴリズム「RL-NewReno」を提案・実装し、その性能評価を行った。RL-NewReno については、異なる行動空間を組み込んだ RL-NewReno-SS と RL-NewReno を設計・実装し、比較実験を実施した。その結果、行動空間にスロースタート挙動を含めた RL-NewReno-SS が優れた性能を示し、既存の NewReno に比べて平均スループットが最大で約 2% 向上することを確認した。

本研究では、主に行動空間の設計に着目したが、今後は報酬設計やモデルの改良を検討する予定である。また、CUBIC や BBR などの帯域を積極的に利用する輻輳制御アルゴリズムとの競合時の公平性評価や、無線通信など環境変化が頻繁に発生するネットワークにおける性能評価を実施する予定である。

文　献

[1] T. Zhang and S. Mao, “Machine learning for end-to-end congestion control,” IEEE Commun. Mag., vol.58, no.6, pp.52–57, 2020.

[2] W. Li, F. Zhou, K.R. Chowdhury, and W. Meleis, “QTCP: Adaptive congestion control with reinforcement learning,” IEEE Transactions on Network Science and Engineering, vol.6, no.3, pp.445–458, 2019.

[3] K. Winstein and H. Balakrishnan, “TCP ex machina: computer-generated congestion control,” Proceedings of the ACM SIGCOMM 2013, pp.123–134, 2013.

[4] P. Gaw lowicz and A. Zubow, “ns-3 meets OpenAI gym: The playground for machine learning in networking research,” Proc. ACM Int’l Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MWIM’19), pp.113–120, 2019.

[5] H. Yin, P. Liu, K. Liu, L. Cao, L. Zhang, Y. Gao, and X. Hei, “ns3-ai: Fostering artificial intelligence algorithms for networking research,” Proc. ACM 2020 Workshop on ns-3, pp.57–64, 2020.

[6] G.F. Riley and T.R. Henderson, “The ns-3 network simulator,” pp.15–34, Springer, Berlin, Heidelberg, 2010.