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
IA2023-44(2023-12)

超低遅延サービスL4Sを用いた 機械学習利用型輻輳制御の性能向上手法の提案

青木　一真†　三橋　力麻††　飯田　勝吉†††　高井　昌彰†††

† 北海道大学 工学部 情報エレクロニクス学科 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　近年、多様な IoT デバイスが幅広く普及しているが、IoT デバイスは一般に計算能力が低いため、高度な計算を必要とする TCP の輻輳制御をデバイス自身で実行することが困難と言える。この課題を解決する一つの方式として、高性能な遠隔サーバを利用して輻輳制御の計算を行わせる方法が考えられる。しかし、IoT デバイスと遠隔サーバ間の通信遅延が大きいと、時々刻々と変わるトラフィック状況に追随することができず、結果として輻輳制御の性能（スループット、ロス率、公平性など）が低下する。本研究では、IoT デバイスと遠隔サーバ間の通信遅延を可能な限り低減させるため、2023 年に RFC9330 として標準化された超低遅延サービス L4S を用いた機械学習利用型輻輳制御の性能向上手法を提案する。

キーワード　IoT, 輻輳制御，遠隔制御，機械学習，RL-TCP, L4S.

A proposal on performance improvement method of machine learning-based congestion control using L4S ultra-low latency services

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

† Laboratory of Advanced Information Network, Division of Computer Science and Information Technology,
Department of Electronics and Information Engineering, School of 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　In recent years, a wide variety of IoT devices have been deployed, but their low computing power makes it difficult to implement TCP congestion control, which requires advanced computation. One possible solution is to use high-performance remote servers to perform congestion control calculations. However, longer communication latency between IoT devices and remote servers might cause performance degradation, such as lower throughput, higher loss rates, and unfairness index. In this research, we propose a method of performance improvement based on machine learning-based congestion control between IoT devices and remote servers using a ultra-low latency communication service called L4S, standardized as RFC9330 in 2023.

Key words　IoT, Congestion control, remote control, machine learning, RL-TCP, and L4S.

1. はじめに

近年、多様な IoT デバイスが幅広く普及しているが、IoT デバイスは一般に計算能力が低いため、高度な計算を必要とする TCP の輻輳制御をデバイス自身で実行することが困難と言える。IoT デバイスのように性能に限りがあるハードウェアでの利用を想定した軽量な輻輳制御アルゴリズム [1]〜[6] の開発も進んでいるが、IoT デバイスの小型化、軽量化、省電力化が続いているなかにあっては、十分な性能の輻輳制御を期待することは難しいといえる。この課題を解決する一つの方式として、高

性能な遠隔サーバを利用して輻輳制御の計算を行わせる方法が考えられる。近年では、機械学習を応用した輻輳制御 [7]〜[16] が数多く登場していることから、IoT デバイスと遠隔サーバの連携によって、より高性能な輻輳制御の実現が期待できる。しかしながら、機械学習を利用した輻輳制御にはいくつかの解決すべき課題がある。例えば、IoT デバイスと遠隔サーバ間の通信遅延が大きいと、時々刻々と変わるトラフィック状況に追随することができず、結果として輻輳制御の性能（スループット、ロス率、公平性など）が低下することになる。

本研究では、IoT デバイスと遠隔サーバ間の通信遅延を可能な限り低減させることを目的として、2023 年に RFC9330 として標準化された超低遅延サービス：Low Loss, Low Latency, Scalable Throughput (L4S) [17] を用いることで、機械学習利用型輻輳制御の性能向上手法を提案する。提案手法の全体図を図 1 に示す。図の中央に示した IoT 機器は、通信 1 を経由して通信相手と接続している。このとき、IoT 機器は輻輳制御を遠隔サーバに依頼し、遠隔サーバは機械学習の機能を使ってその状況に適した輻輳制御を計算して結果を返す。この IoT 機器と遠隔サーバのやり取りが円滑に行われることを目的として、通信 2 に対して L4S を適用する。

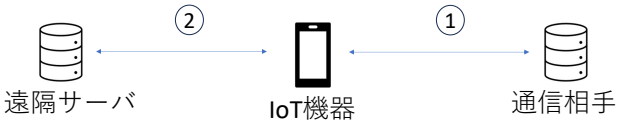


図 1　提案手法の全体図

予備実験では、ns3 を用いて L4S を実装し、L4S パケットの優先制御が有効に機能していることを確認した。また、FIFO 使用時の実行結果と比較することで、低キューイング遅延が実現できていることを明らかにした。

本論文のこれ以降の構成については、2 章で L4S、3 章で関連研究に関して述べた後、4 章で提案アーキテクチャ、5 章で予備実験の内容を説明する。また、6 章では、本論文をまとめたうえで、今後の課題を述べる。

2. L4S

本研究で用いる L4S とは Low Latency, Low Loss, Scalable throughput の略であり、低遅延、低損失、スケーラブルなスループットを実現するための仕組みである。L4S は、パケットに優先順位を付けることによって、ルータのバッファでパケットが待たされることで発生するキューイング遅延を低減させる。L4S を適用したネットワークでは、キューイング遅延を数ミリ秒以下に抑えることが可能となる。ネットワークに流れるパケットが L4S であるか否かは ECN(Explicit Congestion Notification) で決めることになり、L4S 対応のパケットは ECT ビットを 1 としている。L4S の輻輳制御は、遅延とパケットロスに基づいて送信レートの調整を行う古典的な輻輳制御アルゴリズムとは異なり、ECN マーキングを行うことで低遅延の通信を維持することとなる。

ネットワーク上のルータは、L4S パケットと古典的な輻輳制御アルゴリズムで制御されるパケット (以下、Classic パケット) の両方を協調して処理する必要があるが、その仕組みは Dual Queue Coupled AQM [18] を用いて実現する。Dual Queue Coupled AQM では、L4S パケットのためのキューと Classic パケットのためのキューの 2 つのキューを持つ。バッファに流れてきたパケットは L4S 対応か Classic 対応かを分類された後、専用のキューに流れる。各キューにはそれぞれ AQM が用意されており、ネットワークの状態に応じて 2 つの AQM 同士が連携を取ることで、L4S パケットと Classic キューの公平性を保っている。どちらのパケットを排出するかについてはスケジューラが決定するが、基本的には L4S パケットが優先して排出される。なお、本研究の実験で使用する ns3 [19] においては Dual Queue Coupled AQM が実装されていないため、その代替策として FqCoDel [20] を使用する。

3. 関 連 研 究

ここ数年を振り返ると、機械学習を用いたネットワーク輻輳制御に関連して、多くのサーベイ論文が発表されている [8]〜[12]。これらの調査結果は、機械学習を用いたネットワーク輻輳制御が多くの研究の研究者に注目されており、活発に議論が行われている研究分野であることを示している。

具体的な研究報告に目を向けると、Li ら [7] は、Q 学習 [21] を利用した輻輳制御を提案した。メモリに制約のあるデバイスで Q テーブルの保存に関数近似を用いることで、大幅にメモリ使用量を削減し、関数近似を用いない場合と比較して同等の結果が得られたことを示した。Sacco ら [13] は、強化学習に基づくトランスポートプロトコルである Owl を提案した。Owl はエンド・ツー・エンドの特徴とネットワーク信号から適切な輻輳ウィンドウを選択することで高い通信性能を実現した。Molia と Kothari [14] は、パケットロスの原因を予測するためのトランスポート層のソリューションである TCP-RLLD（TCP with Reinforcement Learning based Loss Differentiation）を提案した。TCP のデフォルトでは、どのようなロスも輻輳ロスとして考慮されるが、TCP-RLLD ではそれを無効にし、不必要な伝送レートの低下を防ぐ仕組みとした。Kong ら [15] は、バッファ不足のボトルネックリンクを持つ有線ネットワーク向けに、LP(loss predictor) ベースの TCP 輻輳制御 (LP-TCP) と RL(reinforcement learning) ベースの TCP 輻輳制御 (RL-TCP) を設計し、その性能を評価した。Yin ら [16] は、NS3-AI を提案した。また、RL-TCP を使って高性能な機械学習による輻輳制御アルゴリズムを実装した。

これまでに述べた機械学習を用いた輻輳制御アルゴリズムについては、機械学習の機能を活用するために計算資源を多く消費するが見込まれる。本研究では高性能な遠隔サーバを利用して輻輳制御の計算をすることで、IoT デバイスでも機械学習型の輻輳制御アルゴリズムを利用することを目指す。その際、IoT デバイスと遠隔サーバの間のネットワークに L4S を使用することで、輻輳制御の性能低下を回避する。

4. 提案ネットワークアーキテクチャとその性能評価モデル

1 節で述べたように，きわめて低性能な IoT デバイスで適切な輻輳制御を実施するために、複雑な輻輳制御処理を遠隔地にあるサーバに担当させる方法を検討する．提案アーキテクチャを図 2 に示す．図 2 の右側にあるダンベル網において，リモートサーバが制御する IoT デバイスの TCP フローと競合 TCP フロー間で公平で適切な資源配分を目指す．左側にあるダンベル網では，遠隔制御のための通信と競合 TCP フローが混在している．このような輻輳制御を遠隔制御で実現する際には，一般に通信遅延を小さくすることが求められる．そこで本研究では，図の左側のダンベル網に L4S を導入し，遠隔制御の通信を競合フローに比べて優先させる．なお，TCP 輻輳制御の遠隔制御には強化学習を利用する RL-TCP を用いる．

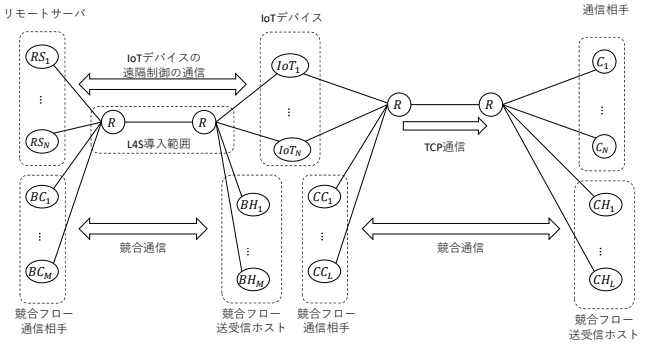


図 2　提案ネットワークアーキテクチャ

提案ネットワークアーキテクチャにおいて、L4S の導入が図 2 の右側のダンベル網において異種フロー間で公平で適切な資源配分の実現可能性を定量的に評価するため、ns3 を用いて性能評価を実施する。その際、ns3 の世界におけるアプリケーションは、ノード間で意図したトラフィックを流すだけであり、何かのタスクをこなすためにトラフィックを流すことができない。そのため、RS_N と IoT_N との間では輻輳制御にかかわる情報を実際にやり取りすることができない。そこで、本研究では FTP ライクな大量のデータ送信を模擬する BulkSendHelper を利用し、以下の手順で仮想的にデータの受け渡しを行う。

（1） RS_N が IoT_N からパケットを受信後、状態を共有メモリに書き込む (C++ から Python へ)。

（2） 状態の書き込み後，python 側で共有メモリを読み込み、ns3-ai の RL-TCP を用いて次の行動を計算する。

（3） IoT_N が RS_N からパケットを受信後、python 側で共有メモリに計算した次の行動を書き込む。

（4） C++ 側で共有メモリから行動を読み込み (Python から C++ へ)。それを用いて IoT_N は TCP 通信を行う。

また、右側のダンベル型網では、IoT_N が C_N と TCP 通信を行う。この通信が遠隔サーバによる制御対象である。なお、CC_L と CH_L がこの通信と同じボトルネックリンクを介して TCP 通信 (競合通信) を行う。

5. 予 備 実 験

前節で述べた提案ネットワークアーキテクチャの評価は ns3-ai を改造する必要があり、一度に実現することが困難であった。そこで、本格的な性能評価環境を構築する前の予備実験として、図 2 の左側のダンベル網と右側のダンベル網を分離した実験を行う。まず、左側のダンベル網単体で IoT デバイスの輻輳制御のための制御通信にかかる遅延時間を計測する。具体的には、左側のダンベル網を取り出したシミュレーション環境を図 3 とし、図 3 において L4S により制御通信を優先した場合とそうでない場合での平均 RTT を計測する。次に、その計測結果を用いて図 2 のダンベル網の性能評価を行う（図 4)。

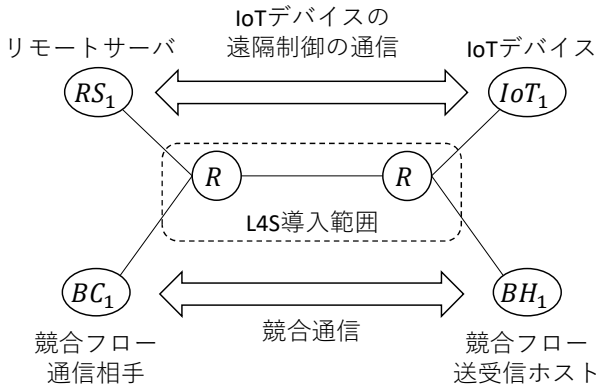


図 3　1 段階目のシミュレーション評価：リモートサーバまでの RTT 計測

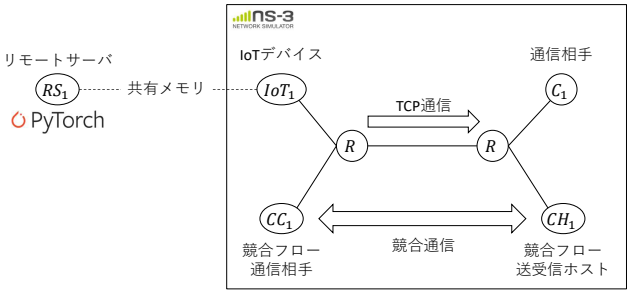


図 4　2 段階目のシミュレーション評価：リモートサーバによる輻輳制御の評価

次に具体的な実装を記載する。本研究ではネットワークシミュレータ ns3 [19] によるシミュレーション実験を行ない、さらに RL-TCP で利用する機械学習の実装は Python で行う。そして，Python と ns3(C++) の間の相互接続には ns3-ai [16] を用いる。ns3-ai は Python 環境と C++ 環境で共有メモリを用いることで相互にデータの受け渡しを行うために、Gymnasium を利用する Gym インターフェース [22] と、Python 環境と C++ 環境で共有データに直接アクセスするメッセージインターフェースを提供している。本研究では Gym インターフェースよりも扱いやすく、シリアライズ/デシリアライズの時間を節約することが可能なメッセージインターフェースを使用することとする。

また、機械学習型輻輳制御については、ns3-ai に組み込まれているである RL-TCP を用いる。RL-TCP には深層強化学習を用いた輻輳制御が実装されており、エージェントは ϵ-greedy アルゴリズムに従って輻輳ウィンドウサイズとスロースタート閾値を選択する。RL-TCP では Python と C++ の相互接続を行いデータの受け渡し、Python による学習ををを行うタイミングを一定時間ごとに行う仕様にしている.

実験に用いた計算機とソフトウェアの環境は、CPU: Intel Core i9-12900, Memory: 64GB, GPU: Nvidia RTX 3080, OS: Ubuntu22.04LTS, ns-3.38, Python 3.10.12, Pytorch 2.1.1, CUDA 12.1 である。

次に、図 3 のシミュレーション評価の詳細を説明する。このトポロジーにおいて RS_1 の平均 RTT の計測を行なう。このトポロジーでは、左側のノードから右側のノードに BulkSendHelper を用いてパケットを送信しており、RS_1 の輻輳制御は ns3-ai の RL-TCP で実装されている Python で書かれたコードを利用し、BC_1 の輻輳制御は CUBIC で行う。なお、Python による輻輳制御は 0.1 秒ごとに行なう。また、L4S 使用時は AQM として FqCoDel を，L4S 不使用時は通常の FIFO を使用する．L4S 使用時には RS_1 が ECT1 パケットとなり，L4S による優先制御を受ける。

最後に、図 4 による 2 段階目のシミュレーション評価の詳細を説明する。このトポロジーでは、RS_1 の輻輳制御は RL-TCP (ns3-ai) の Python コードを利用し、BC_1 の輻輳制御は CUBIC で行う。その際、リモートサーバでの輻輳制御を模擬するため、RL-TCP の強化学習による輻輳制御は、輻輳制御呼び出し後に即座に制御を行わず、１段階目の評価（図 3）で計測した RS_1 の平均 RTT の時間を待たせることにした。なお、図 3 と同様に、図 4 でも左側のノードから右側のノードに BulkSendHelper を用いてパケットを送信する。ボトルネックリンクでのパケット廃棄は FIFO による Tail Drop に従うものとした。.

以上のシミュレーション設定は、簡易的に図 2 における N, M, L が 1 のときのシミュレーション環境を表現することを意図している (図 5)。

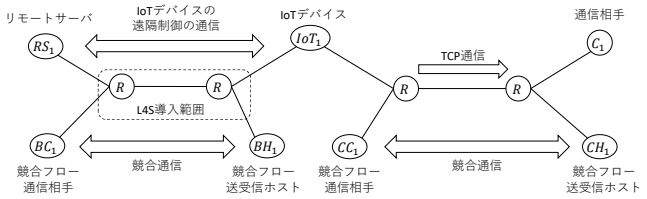


図 5　予備実験の全体像

5.1　シミュレーションパラメータ

予備実験で用いたパラメータを表 1 に示す。なお、表のアクセスリンク 2 は、図 3, 図 4 のルータと競合フロー送受信ホスト間のアクセスリンクを、アクセスリンク 1 はそれ以外のアクセスリンクを示す。

5.2　シミュレーション結果

まず、１段目のシミュレーション評価（図３）の RS_1 の平均 RTT の結果を表 2 に、輻輳ウィンドウサイズ、RTT、キューイング遅延を図 6 (L4S 有効時)、図 7 (L4S 無効時) に示す。

表 1　シミュレーションパラメータ

パラメータ	１段目（図 3）	２段目（図 4）
ボトルネックリンク遅延	0.01ms	0.01ms
ボトルネックリンク帯域	10Mbps	10Mbps
アクセスリンク 1 遅延	10ms	10ms
アクセスリンク 1 帯域	100Mbps	100Mbps
アクセスリンク 2 遅延	9.5ms	9.5ms
アクセスリンク 2 帯域	100Mbps	100Mbps
競合通信の輻輳制御アルゴリズム	CUBIC	CUBIC
送受信者のバッファサイズ	500KB	500KB
シミュレーション実行時間	200s	200s

表 2　RS_1 の平均 RTT：１段目（図 3）

	L4S 使用時	FIFO 使用時
平均 RTT	65ms	105ms

次に、図 4 にてシミュレーションを行った時の輻輳ウィンドウサイズ、RTT を図 8、図 9 に示す。

5.3　考　　察

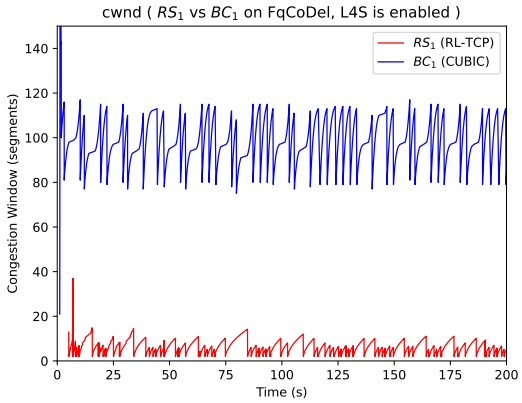
図 6、図 7 を比較すると、まず CUBIC の輻輳ウィンドウサイズサイズについては L4S 使用時の方が低い値を推移していることが分かる。これは、L4S 使用時、RS_1 のパケットが優先されているため、CUBIC が輻輳ウィンドウサイズを上げにくくなっているからであると考えられる。

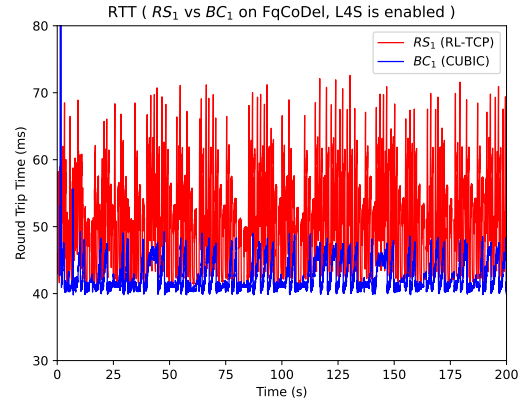
また、キューイング遅延については、L4S 使用時、RS_1 のパケットのキューイング遅延が主に 0~2ms の範囲で推移しているのに対し、CUBIC のパケットのキューイング遅延は、主に 0~10ms の範囲で推移しており、RS_1 のパケットが L4S の制御を受けることで低キューイング遅延を実現していることが分かる。

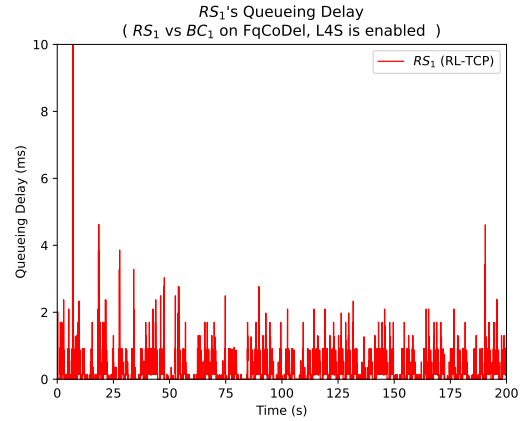
次に、２段目の性能を図 8 と 9 により評価する。L4S 有効時、無効時、どちらも CUBIC が大きな資源を獲得しているため、RL-TCP は適切なスループットを獲得できていない。しかし、１段目の RTT により設定した遅延時間は１段目で L4S を有効にした図 8 の方が短く、そのためパラメータによっては性能差が生じる可能性がある。いずれにしても、CUBIC と競合時に適切にスループットを獲得可能なアルゴリズムのさらなる検討が必要と言える。

6.　お わ り に

IoT デバイスは一般に計算能力が低いため、高度な計算を必要とする TCP の輻輳制御をデバイス自身で実行することが困難と言える。この課題を解決する一つの方式として、高性能な遠隔サーバを利用して輻輳制御の計算を行わせる方法が考えられるが、IoT デバイスと遠隔サーバ間の通信遅延が大きいと輻輳制御の性能が低下することになる。







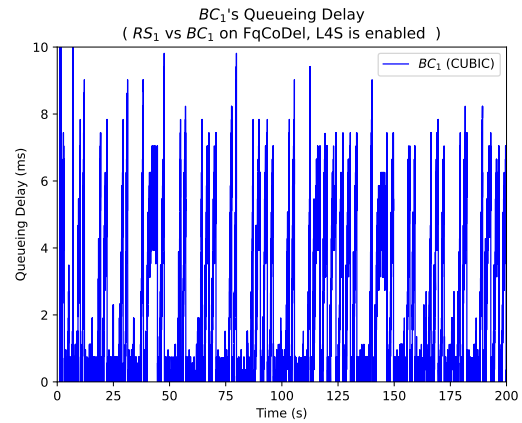
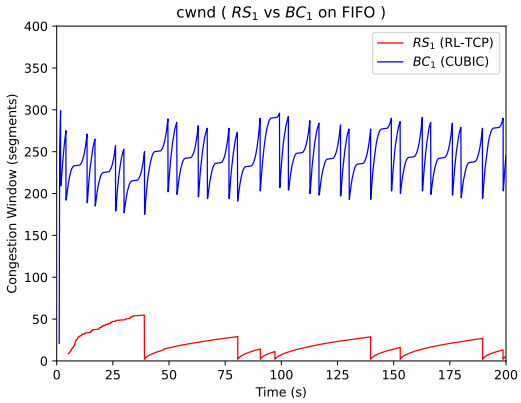
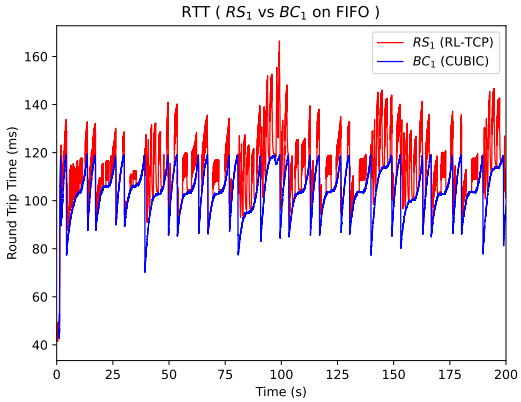
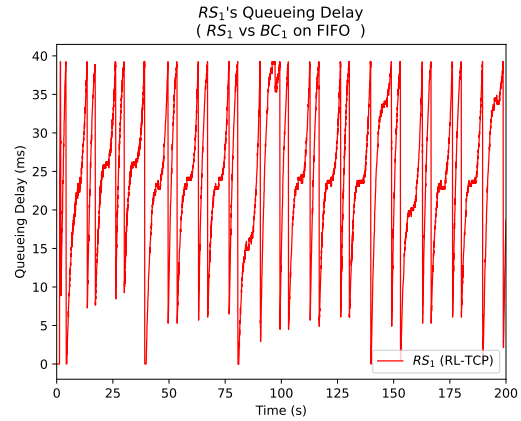


図 6　1 段目のシミュレーション結果：L4S 有効







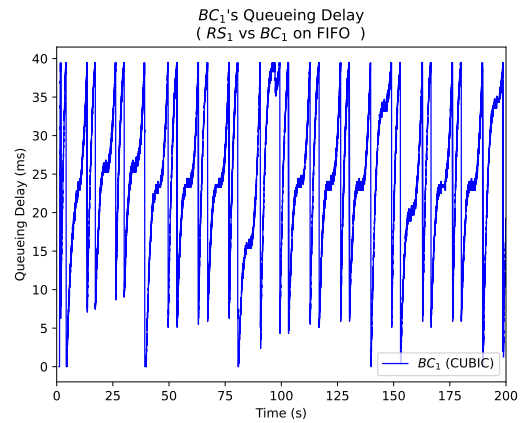


図 7　1 段目のシミュレーション結果：L4S 無効

本研究では、IoT デバイスと遠隔サーバ間の通信遅延を可能な限り低減させるため、超低遅延サービス L4S を用いた機械学

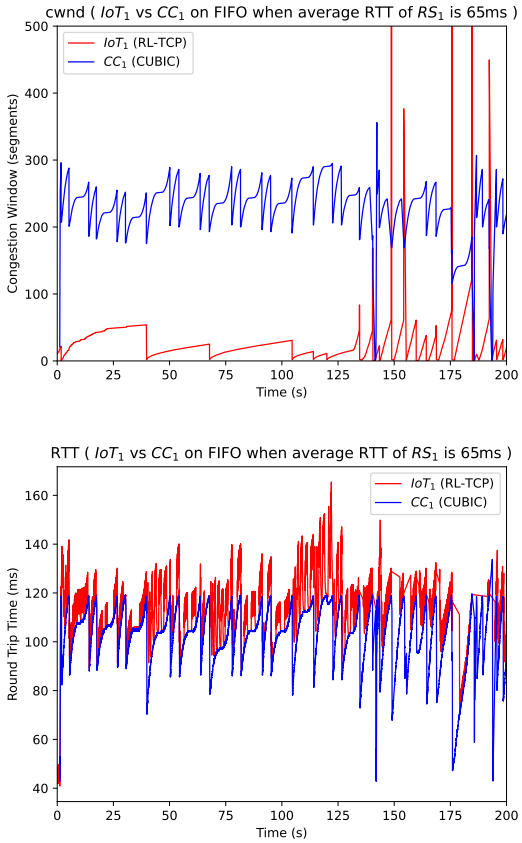


図 8　２段目のシミュレーション結果：RL-TCP の遅延=65ms（１段目で L4S 有効時）

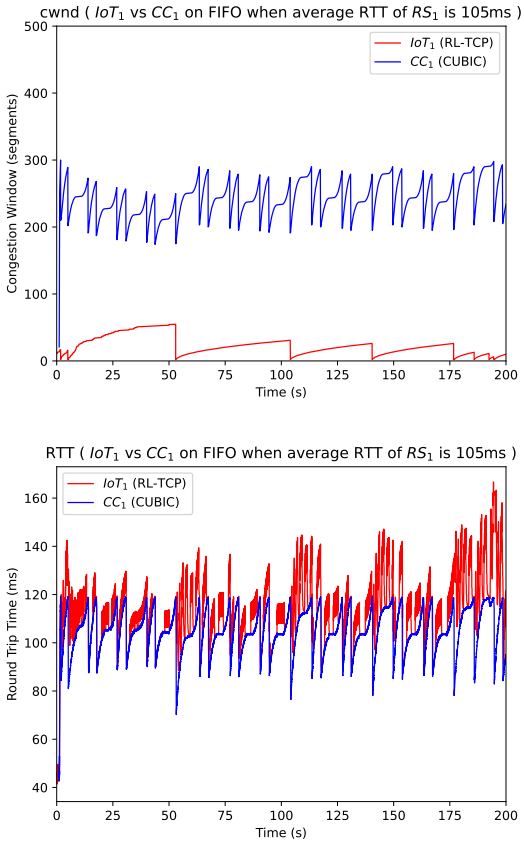


図 9　２段目のシミュレーション結果：RL-TCP の遅延=105ms（１段目で L4S 無効時）

習利用型輻輳制御の性能向上手法を提案した。予備実験では、輻輳制御の遠隔制御部分（１段目）とそれを用いた輻輳制御部分（２段目）に分け、シミュレーション評価を行った。１段目の評価により、L4S の有無が制御パケットの RTT に与える影響を調査し、その結果を用いて２段目のシミュレーションを行った。今回の実験では、CUBIC と遠隔制御 RL-TCP が競合すると、CUBIC が資源を占有し、RL-TCP がスループットをほとんど獲得できず、L4S を有効にしても、その改善度合いは顕著ではないことが明らかになった。CUBIC などの近代的な輻輳制御方式と、適切に資源配分が可能な遠隔制御型輻輳制御について、今後検討を進める予定である。

文　　献

[1] H.A. Khalek and L. Mhamdi, “Light-weight congestion control in constrained IoT networks,” Proc. IEEE Global Communications Conference (GLOBECOM 2022), pp.6265–6270, 2022.

[2] V. Diniesh, E. Shanjeev, V. Saran, B. Tamilmani, et al., “On minimizing TCP traffic congestion control in vehicular internet of things,” Proc. IEEE Int’l Conference on Electronics and Sustainable Communication Systems (ICESC 2022), pp.446–451, 2022.

[3] N. Makarem, W.B. Diab, I. Mougharbel, and N. Malouch, “On the design of efficient congestion control for the constrained application protocol in IoT,” Computer Networks, vol.207, p.108824, 2022.

[4] V.K. Jain, A.P. Mazumdar, P. Faruki, and M.C. Govil, “Congestion control in internet of things: Classification, challenges, and future directions,” Sustainable Computing: Informatics and Systems, vol.35, p.100678, 2022.

[5] T.N. Pham, D.H. Hoang, and T.T.D. Le, “Fuzzy congestion control and avoidance for CoAP in IoT networks,” IEEE Access, vol.10, pp.105589–105611, 2022.

[6] L. Ma, X. Liu, H. Wang, Y. Zhou, and X. Deng, “Congestion control for internet of things based on priority,” International Journal of Control, Automation and Systems, vol.20, no.4, pp.1154–1165, 2022.

[7] W. Li, F. Zhou, W. Meleis, and K. Chowdhury, “Learning-based and data-driven TCP design for memory-constrained IoT,” Proc. IEEE Int’l Conference on Distributed Computing in Sensor Systems (DCOSS2016), pp.199–205, 2016.

[8] R. Boutaba, M.A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O.M. Caicedo, “A comprehensive survey on machine learning for networking: evolution, applications and research opportunities,” Journal of Internet Services and Applications, vol.9, no.1, pp.1–99, 2018.

[9] F. Tang, B. Mao, Y. Kawamoto, and N. Kato, “Survey on machine learning for intelligent end-to-end communication toward 6G: From network access, routing to traffic control and streaming adaption,” IEEE Communications Surveys & Tutorials, vol.23, no.3, pp.1578–1598, 2021.

[10] M. Polese, F. Chiariotti, E. Bonetto, F. Rigotto, A. Zanella, and M. Zorzi, “A survey on recent advances in transport layer protocols,” IEEE Communications Surveys & Tutorials, vol.21, no.4, pp.3584–3608, 2019.

[11] Y. Cheng, J. Geng, Y. Wang, J. Li, D. Li, and J. Wu, “Bridging machine learning and computer network research: A survey,” CCF Transactions on Networking, vol.1, pp.1–15, 2019.

[12] P. Anitha, H. Vimala, and J. Shreyas, “Comprehensive review on congestion detection, alleviation, and control for IoT networks,” Journal of Network and Computer Applications, p.103749, 2023.

[13] A. Sacco, M. Flocco, F. Esposito, and G. Marchetto, “Owl: Congestion control with partially invisible networks via reinforcement learning,” Proc. IEEE Conference on Computer Communications (In-

focom2021), pp.1–10, 2021.
[14] H.K. Molia and A.D. Kothari, “TCP-RLLD: TCP with reinforcement learning based loss differentiation for mobile adhoc networks,” Wireless Networks, vol.29, no.5, pp.1937–1948, July 2023.
[15] Y. Kong, H. Zang, and X. Ma, “Improving TCP congestion control with machine intelligence,” Proc. ACM Workshop on Network Meets AI & ML, pp.60–66, 2018.
[16] 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.
[17] B. Briscoe, K. De Schepper, M. Bagnulo, and G. White, “RFC 9330: Low latency, low loss, and scalable throughput (L4S) internet service: Architecture,” 2023.
[18] K. De Schepper, O. Albisser, O. Tilmans, and B. Briscoe, “Dual queue coupled AQM: Deployable very low queuing delay for all,” 2022. arXiv preprint arXiv:2209.01078.
[19] G.F. Riley and T.R. Henderson, “The ns-3 network simulator,” pp.15–34, Springer, Berlin, Heidelberg, 2010.
[20] T. Hoeiland-Joergensen, P. McKenney, D. Taht, J. Gettys, and E. Dumazet, “RFC 8290: The flow queue CoDel packet scheduler and active queue management algorithm,” 2018.
[21] C.J. Watkins and P. Dayan, “Q-learning,” Machine learning, vol.8, pp.279–292, 1992.
[22] 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.