Advances in Nonlinear Variational Inequalities
ISSN: 1092-910X
Vol 28 No. 7s (2025)

Delay Driven Machine Learning Scheme for TCP Data Flow Performance Enhancement

[1]Keerti Mishra, [2]Nitin Jain, [3]Saurabh Bhutani
[1,2,3]Electronics and Communication Engineering, School of Engineering, BBD University,
Lucknow, India
[1]E-mail: keertim14@gmail.com

Article History:

Received: 18-12-2024

Revised:28-1-2025

Accepted:6-2-2025

Abstract:

Wireless networks with a substantial Bandwidth Delay Product (BDP) often use the Transmission Control Protocol (TCP) for conventional data delivery. However, TCP expands the congestion window dramatically, often resulting in ineffective bandwidth consumption. Furthermore, its congestion monitoring system adds to significant packet loss, a problem increased by improved network devices and the growing Internet. This paper presents an intelligent solution to congestion control that uses machine learning models to automatically change the number of virtual parallel streams. This approach improves congestion control by adjusting delays depending on fluctuations in the Cwnd. MATLAB-based simulations show that the suggested technique outperforms existing algorithms in terms of bandwidth efficiency.

Keywords: congestion control, delay driven mechanism, machine learning, TCP, wireless communication ipsum.

1. Introduction

Congestion control methods are critical for ensuring network stability, balanced resource allocation, fast throughput, and minimal switch queuing delays. Numerous alternatives have been offered; nonetheless, TCP remains the dominant Internet and LTE communications protocol, with the majority of congestion control systems based on TCP [2]. Despite its extensive use, research has revealed limits in dynamic contexts that differ from those for which it was initially developed [3–7]. These issues are visible in cellular and wireless networks, where TCP, particularly Cubic—the default on many devices—frequently misinterprets stochastic packet losses as congestion, resulting in performance reduction [7]. To address this, researchers have investigated novel congestion management techniques based on a domain-specific design philosophy, in which solutions are adapted to specific network circumstances, utilizing their distinct properties to improve performance [6]-[9]. The difficulty of properly updating the C_{wnd} in resource-constrained networks, such as wireless networks and IoT, is further complicated by intrinsic constraints in bandwidth, computing power, and battery life, as well as their highly dynamic nature [10]-[12]. TCP's deterministic nature contributes to Cwnd synchronization difficulties and increases congestion losses,

especially in wireless multi-hop networks with node mobility that constantly changes communication pathways [11], [13]. To solve these issues, numerous TCP versions have been proposed, including PCC [14] and Copa [15]. However, these methods are based on fixed-rule procedures, which frequently fail to adapt to quickly changing network conditions. To address this shortcoming, machine learning learning-based transport protocol offered here that dynamically optimizes congestion window updates.

Unlike previous machine learning-based transport protocols, this technique trains the model while also using existing mechanisms to calculate the next C_{wnd} value. Several transport protocols have been devised, both with and without machine learning, to construct network-aware solutions [8], [18], [19]. However, contemporary systems have yet to properly leverage network intelligence. Optimally updating C_{wnd} improves throughput and fairness while reducing packet loss and delay. The proposed transport protocol achieves these goals by learning from various end-to-end and in-network indicators. The significant contributions are as follows: Designed and implemented a novel congestion management protocol that uses partial network knowledge to train a machine learning model, resulting in improved network performance above previous techniques [20, 21]. The suggested model calculates the next C_{wnd} value, which is a key and highly dynamic parameter in reliable telecommunications. To determine its efficacy, an exhaustive study was performed, comparing the proposed model to various transport protocol implementations. Some of them were created expressly for wireless networks [6, 18], while others [22]-[25] are extensively used across a variety of platforms. The performance study, which is based on emulations of real-world traces from Verizon and T-Mobile, as well as deployment on the GENI testbed [26], shows that the proposed model consistently increases bandwidth and decreases latency across numerous scenarios. Furthermore, we investigate the parameters of machine learning models and assess their fairness performance, demonstrating that it is less aggressive than other transport protocols. Furthermore, we examine the effects of partial network visibility and show that our model can function successfully with few or no in-network congestion signals. Furthermore, it is demonstrated that the sender may learn effective congestion window adjustment tactics across a variety of network settings and dynamically adapt to changing circumstances. The remainder of the paper is organized as follows: Section II examines related studies and crucial processes for comparison. Section III explains our framework and its key features. Section IV describes our issue formulation and procedure design, followed by a result analysis in Section V. Finally, Section VI summarizes the article and offers future research areas.

2. Related Work

Congestion control and avoidance have received a lot of attention in the literature because of their importance in maintaining reliable data transfer. Recently, machine learning-based techniques have demonstrated potential in addressing the difficulty of optimum congestion window prediction. This section focuses on how these solutions differ from our suggested protocol. Congestion control is a basic component of TCP, which has resulted in various advancements. Some notable examples include TCP Vegas [24], Compound [27], Fast [28],

ExLL [29], BBR [25], and Data Center TCP (DCTCP) [8]. Unlike previous systems, which modify the Cwnd based on packet loss, BBR uses round-trip time (RTT) and delivery rate metrics to govern data transmission speed. While this helps BBR alleviate the bufferbloat issue, it frequently exceeds connection capacity, resulting in significant queuing delays [18]. Other protocols, such as Compound [27] and Fast [28], prioritize packet loss reduction but rely on preset fixed rules to control network circumstances. As a result, struggle in networks that need fast adaptation. In contrast, our system overcomes this constraint by using machine learning to dynamically forecast the optimal Cwnd update at each transmission event. Machine Learning (ML) has lately acquired popularity for tackling a variety of network operational difficulties [30]. Many of these systems are intended for resource-constrained networks, such as IoT [10] and WANETs [11], [13], [31], whilst others cater to a larger variety of architectures [5], [14], [32]. Recent end-to-end congestion control methods, like Remy [5], PCC [14], and PCC-Vivace [33], specify objective functions to optimize decision-making processes on a regular basis. Remy [5], for example, uses offline training under various network situations to establish the best mapping for sender behavior. These mappings are precomputed and stored in a lookup table, thus the system can only adjust to changing network conditions by recalculating the table. In contrast, PCC [14], PCC-Proteus [34], and PCC-Vivace [33] all execute real-time optimizations. PCC flexibly responds to changing network circumstances by constantly looking for the most effective ways to change the transmitting rate. However, these online optimization algorithms are frequently sophisticated, resulting in large delays in precisely calculating network parameters. Copa [15] takes a similar utility-based approach, using a delay-based congestion management algorithm to change the Cwnd based on whether the current transmission rate is above or below a predetermined target rate. This technique allows for quick convergence to fair transmission rates, even in the presence of severe flow fluctuation. Similarly, our protocol implements a utility-based structure while improving flexibility. This enables for specialized optimizations while maintaining utility customization flexible and adaptable to particular requirements. Several transport protocols rely on Explicit Congestion Notification (ECN) to give network-level feedback. For example, DCTCP [8] adjusts ECN's Red Early Drop levels to provide high throughput and burst tolerance while retaining low queue occupancy, hence reducing latency. Other protocols, such RCP [36], XCP [37], and D3 [9], modify switch behavior to offer feedback on transmission rates. More contemporary techniques, such as NATCP [38] and HPCC [39], use switches (or a centralized organization in the case of NATCP) to transmit bottleneck link circumstances. ABC [18] improves ECN by employing explicit accelerate and brake signals, which enable for transmission rate modifications. Swift [40] enhanced communication by introducing congestion control based on network latency. Similar to ABC, the suggested approach makes use of network-level information. Rather than changing packet headers, it gets data from a network controller, such as a measurement agent or SDN controller, which collects device use information. This removes the deployment issues associated with altering packet structures or network hardware. It uses solely client-side updates and a network statistics collector. Unlike ECN-based techniques, which rely on network information for operation, it can function well without network-level feedback.

Furthermore, when network feedback is provided, its insights extend beyond a single bit in the TCP header, providing more detailed information for congestion control.

3. Methodology

The concept of virtual parallelism in TCP is follows the simulation of different parallel data streams to tackle the problem of under-utilization of bandwidth. It performs modification of the mechanism of the Additive Increase Multiplicative Decrease (AIMD) of TCP Reno. When tested for 100 Mbps bottleneck link scenario at RTT of 80 ms, and a 0.02 percent rate of packet loss, MulTCP utilizes the bandwidth effectively that would otherwise unused in TCP Reno. However, when MulTCP operates with a large number of parallel streams (N), it can negatively impact the throughput of TCP Reno flows, indicating poor synchronization and reduced fairness to TCP Reno. The TCP-FIT [24] scheme addresses this problem with dynamic adjustment of the value of N. However, the adjustment is not always timely or accurate, which motivates the development of a new approach—this research proposes a decision tree-based TCP algorithm. In TCP Inigo [25], the RTT of each ACK is used for estimating the level of congestion of network. For every ACK received, the decision scheme follows a specific rule to guide congestion control.

$$RTT_{observed} = RTT_{observed} + 1 \quad (1)$$

$$\text{and if : } RTT > RTT_{observed} + d_{thresh} \quad (2)$$

$$\text{then : } \quad RTT_{S\_late} = RTT_{S\_late} + 1 \quad (3)$$

RTT_{min} is lowest limit of RTT. It is approximate propagation delay of network. $RTT_{observed}$ is observed RTT. RTT_{late} is observed RTT times that is exceeding the RTT threshold. d_{thresh} is the delay threshold of queue. If K is the middle buffer size threshold, and the C representing the middle link bandwidth:

$$D_{thresh} = \quad K / C \quad (4)$$

BFFS: Buffer Free Space Size, CL: Congestion level

Partial Network Availability (PNA): It informs about the available (bandwidth) resources in the network. It indicates the spare capacity of the network, in a similar way to [37]. For any link j, given C its capacity and cr the current traffic rate, we define the spare capacity on such a link, scj, as $(C\text{-}c_r)/C$. Then, given w links on the path between a source and a destination, we define P NA as follows:

$$PNA = \min(sc_1, sc_2, ..., sc_w)$$

PNA is easy to compute and accessible by a vast number of protocols and network measurement applications.

In the proposed work presented in this article a smart decision mechanism approach based on machine learning is used to decide the level of congestion *CL* as the output variable using the value of *RTT, BFSS and PNA* as shown in table 1. It follows the rules that decides the *CL* as high if *RTT* is high (Max) ‘AND’ *BFSS* is low (‘Min’) ‘AND’ *PNA* is low (‘Min’). It means if large delay (*RTT*) is observed or free space in buffer is low then it may be assumed that congestion level is high (‘Max’). Under the case of high *CL* as per the equation (1), (2) and (3) additional wait time (delay) is inserted prior to send next packet. The inserted delay is incremented until the *CL* do not reaches medium (‘Mid’) level. Using the rule based shown in table 1 the machine learning tool under MATLAB software is used to develop three models

known as Decision tree (DT), K-nearest neighbor (KNN), Naive Bayes (NB) and Fuzzy Logic (FL) structure.

TABLE1. RULES FOR PROPOSED MACHINE LEARNING DECISION MODEL BASED CL ESTIMATION UNDER TCP

RTT	BFSS	PNA	*CL*	RTT	BFSS	PNA	CL	RTT	BFSS	PNA	CL
Min	Min	Min	Max	Mid	Min	Min	Max	Max	Min	Min	Max
Min	Min	Mid	Mid	Mid	Min	Mid	Mid	Max	Min	Mid	Max
Min	Min	Max	Min	Mid	Min	Max	Mid	Max	Min	Max	Mid
Min	Mid	Min	Mid	Mid	Mid	Min	Mid	Max	Mid	Min	Mid
Min	Mid	Mid	Mid	Mid	Mid	Mid	Mid	Max	Mid	Mid	Mid
Min	Mid	Max	Mid	Mid	Mid	Max	Mid	Max	Mid	Max	Mid
Min	Max	Min	Min	Mid	Max	Min	Mid	Max	Max	Min	Mid
Min	Max	Mid	Mid	Mid	Max	Mid	Max	Max	Max	Mid	Max
Min	Max	Max	Min	Mid	Max	Max	Min	Max	Max	Max	Max

3.1 Decision Tree (DT)

DT is a non-parametric supervised learning system with a hierarchical tree structure made up of a root node, branches, internal (decision) nodes, leaf internal (decision) nodes. It begins at the root node, with branches leading to internal nodes that evaluate features and generate subsets. The leaf nodes, known as terminal nodes, reflect the dataset's results. A decision tree's flowchart structure improves decision-making while also offering a clear depiction of decisions. It employs a divide-and-conquer technique combined with a greedy search to locate optimal split sites, splitting data iteratively from top to bottom until the majority of records are classified. Smaller trees typically reach pure leaf nodes (single-class data points), whereas larger trees frequently have data fragmentation, raising the danger of overfitting. To avoid this, pruning eliminates branches of minor feature importance, lowering complexity. Cross-validation is used to assess performance, and the CART algorithm selects the optimal split using Gini impurity. Gini impurity assesses the chance of misclassifying a randomly selected data point, with a lower value being preferable. A pure set belonging to a single class has a Gini impurity of 0.

$$\text{Gini Impurity} = 1 - \sum_i (p_i)^2$$

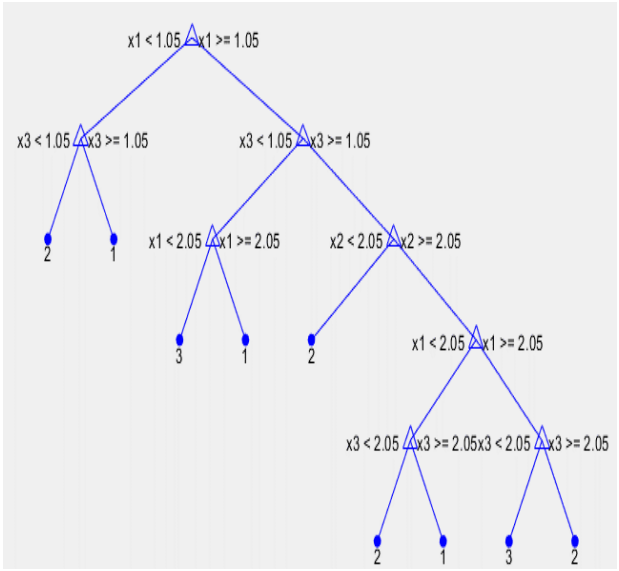

Figure 1: Decision tree structure for DTBTCP

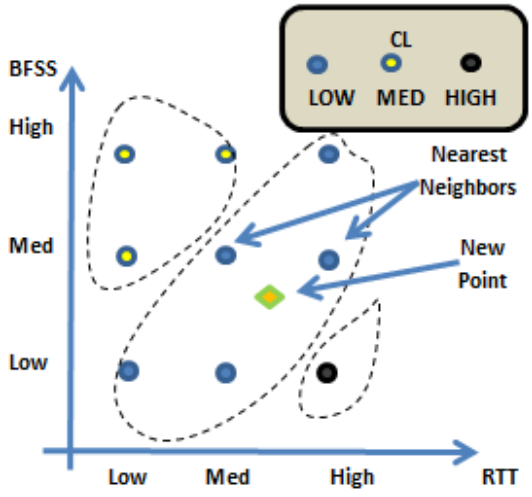

Figure 2: k-NN structure for determining new data point congestion level

The decision tree developed by using rule base of table 1 is shown in figure 1. Here { x_1, x_2, x_3} are input variable {*RTT* , *BFSS*, *PNA*} and the output is CL having three possible class label value as 1,2 or 3 i.e. LOW, MED or HIGH. This decision tree model is developed in MATLAB using Statistics and Machine Learning Toolbox.

3.2 K-Nearest Neighbor (KNN)

K-NN algorithm is a straightforward, supervised technique under machine learning for categorizing new data query on the basis of its resemblance to existing data. It saves the dataset and classifies additional points at runtime without making any assumptions about the underlying data, hence it is non-parametric. In this article, K-NN is used to classify data. CL is divided into three categories: low, medium, and high. Given a new data point with x1 (RTT), x2 (BFSS) and x_3 (PNA), K-NN determines which category it belongs to on its resemblance to previous data. Following steps are followed to develop KNN decision model:

1. Selection of number of neighbors (K).
2. Calculate distance between K neighbors.
3. Choose the K nearest neighbors.
4. Determine the number of data points in each CL category among the K neighbors.
5. Put new data into category with the most neighbors.

3.3 Kernel Based Naive Bayes Classifier

A Naive Bayes classifier is a simple probabilistic type classifier that uses Bayes'. It calculates the probability of a class using independent features, even if they are connected. For example, if RTT is "High" and BFSS is "Low," the classifier may predict the class as "High." Despite assuming independence, Naive Bayes works well with little training data. In this article, we employ a Kernel-based Naive Bayes classifier. Kernels are weighting functions used in nonparametric estimation to estimate the density functions of random variables. Non-parametric estimators make estimates based on all data points, as opposed to parametric estimators, which have a fixed framework. Here X is the input feature set of RTT and BFSS as (x_i= x_1, $x_{2....,}$ x_N) and C is the set of CL level (class label) such that $c_j \epsilon$ {Low, Med, High}.

$$c = \max_{c_j \in C} P(c_j | x_1, x_2 \ldots x_n) \qquad (6.1)$$

$$P(x_1, x_2 \ldots . x_n \mid c_j) = \prod_i P(x_i \mid c_j) \qquad (6.2)$$

$$P(x_i \mid c_j) = \frac{1}{Nh} \sum_{v=1}^{N} K(x_i, x_{iii}) \qquad (6.3)$$

$$K(a, b) = \text{kernel function} = \frac{1}{\sqrt{2\pi}} e^{\frac{(a-b)^2}{2h^2}} \qquad (6.4)$$

$P(x_i|c_j)$ is the probability of the CL equal to level x_i given that the input RTT and BFSS belongs to class label c_j estimated using training data (X,C). Here Kernel is a Gaussian function kernel with mean zero and variance 1, N : number of the input ata X belonging to class j which is equal c_j , x_{iii} is the feature value of the CL in the i^{st} position of the iii^{rd} input X

= (x_{1i}, x_{2i} … x_{Ni}) in class j , and h is a bandwidth used as smoothing parameter. To optimally estimate the conditional probabilities, h is optimized on the basis of training dataset.

3.4 Fuzzy Logic

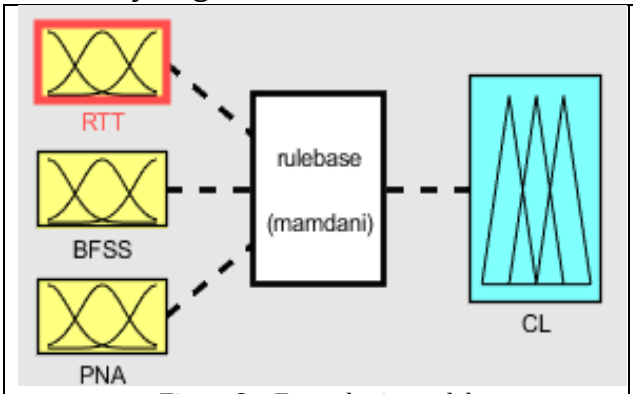

Figure 3a. Fuzzy logic model

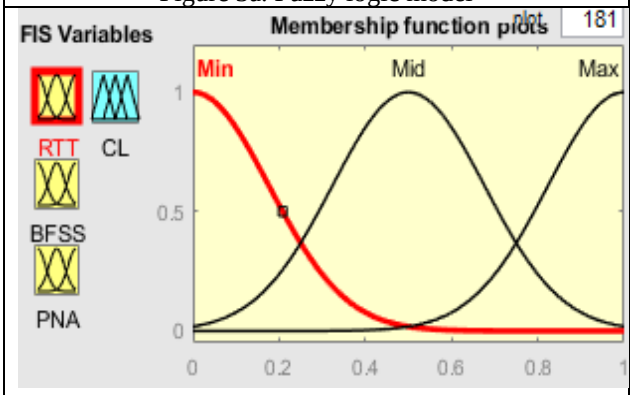

Figure 3b: Membership Function

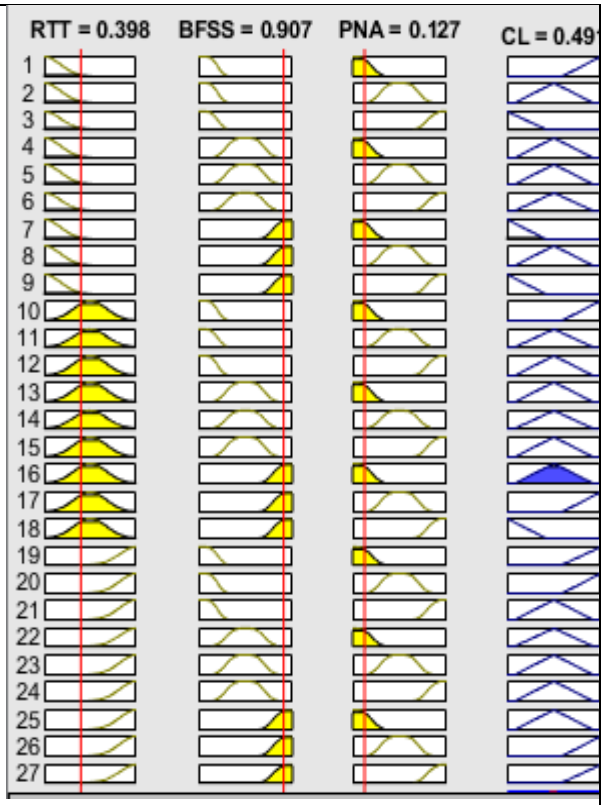

Figure 3c: Rule base fuzzy logic evaluation mechanism.

Fuzzy logic is the simple and straightforward machine learning model based on rule base made by human experience. In this article the fuzzy logic system is developed to decide the congestion level using fuzzy algebra with respect to input value of three inputs as X={RTT, BFSS,PNA}={x_1, x_2, x_3} as shown in figure 3a. The input and output variables are partitioned Gaussian into membership function as shown in figure 3b. In fuzzy logic all the input value is distributed among all membership function (Min, Mid, Max) at specific probability and final output is evaluated as weithed sum of all the fuzzy rules to find most compromising solution as shown in the figure 3c.

4. Results

The implementation is performed on MATLAB software using Instrument Control toolbox and Machine learning toolbox. The TCP protocol functions are used to implement different codes for server port and client for. Two different instances of MATLAB run separately on same computer in parallel mode to create server and client side environment. A standard topology single dumbbell link (figure 4a) is implemented; n represents number of clients

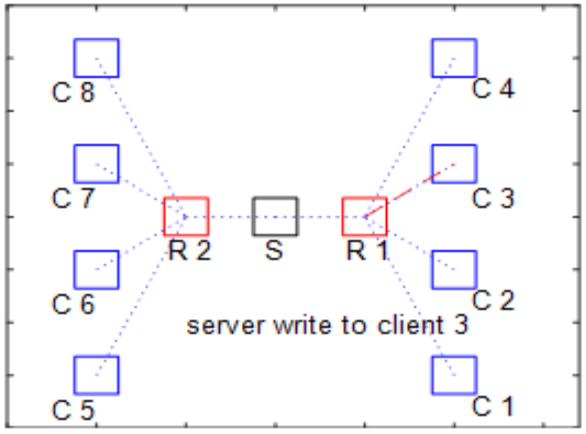

Figure 4a: Data flow instant during writing data from server to client3.

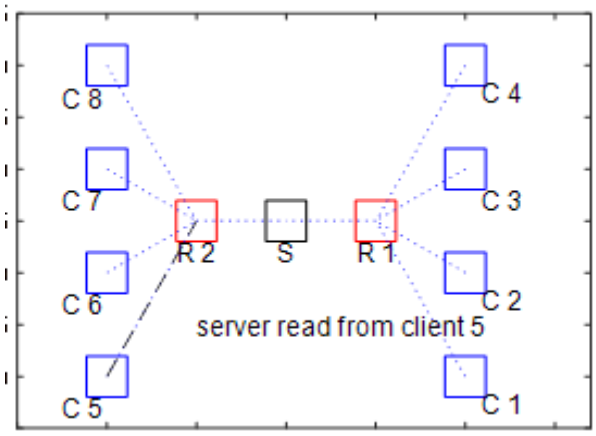

Figure 4b: Data flow instant during reading data from client 5 to server.

Table 2: Results summary for applying different machine learning models

		k-fold	Accuracy	misclassification cost	obs/sec	train time
Decision Tree	cross validation	2	33%	18	6200	0.839
		3	40.70%	16	4000	0.842
		4	33%	18	3400	0.857
		5	44%	15	2600	0.901
	hold out validation	25%	50%	3	3300	0.93
		35%	**55.60%**	4	3000	1.48
		45%	50%	6	3900	0.95
		50%	46%	7	2600	1.187
Kernel Naive Bayes	cross validation	2	44.40%	15	2300	0.966
		3	59%	11	1500	1.136
		4	59.30%	11	1400	1.141
		5	59%	11	890	1.157
	hold out validation	25%	66.6%	2	680	1.176
		35%	**66.70%**	3	2200	1.218
		45%	58%	5	1400	1.274
		50%	54%	6	3500	1.202
KNN	cross validation	2	63%	10	2100	2.256
		3	51.9%	16	1500	2.341
		4	59.3%	11	1500	1.378
		5	55.6%	12	1400	1.329
	hold out validation	25%	**66.7**%	2	2500	1.476
		35%	66.5%	3	2800	1.432
		45%	58.3%	5	4200	1.422
		50%	61.50%%	5	5100	1.4112
Fuzzy Logic	cross validation	2	53%	10	1100	2.526
		3	61.9%	9	2500	1.491
		4	68.3%	2	5200	1.013
		5	51.6%	7	4400	1.025
	hold out validation	25%	56.7%	4	1500	1.261
		35%	46%	7	3800	1.132
		45%	**69.3**%	2	5100	1.312
		50%	51.50%	6	3200	1.041

(Client1 to Client4) at one end (left side) and (Client5 to Clien4) at another end (right side) which send or receive data through the server using sharing of common bottleneck single link of buffer size one BDP. Figure 4a and 4b showing two different events during write operation (packet transmission from server to client 3) and read operation (packet transmission from client 5 to server) through the router R1 and R2. All the machine learning models are developed using machine learning toolbox and integrated with TCP/IP interface under instrumentation toolbox under MATLAB software. The weather data of pressure value imported through csv file is randomly transmitted and received through the dumb bell shape link. Total 40 packets are transmitted over the interface and the accuracy of various ML models under different settings are evaluated and summarized in table 2. The parameter that are set during learning of ML models are number of k-fold validation as {2,3,4 and 5} folds, percent hold out validation (25%, 35%, 45% and 50%) and performance is measured in terms of Accuracy, misclassification cost, observations per seconds (obs/sec) and training time.

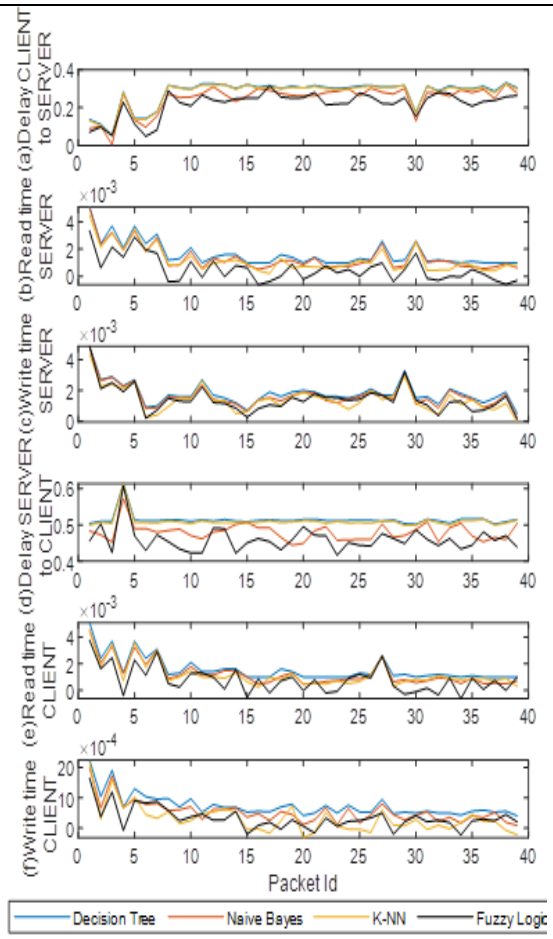

Figure 5: Plots showing performance as (a) Delay (b) Read time (d) delay server to client (e) read time client (f) write time client.

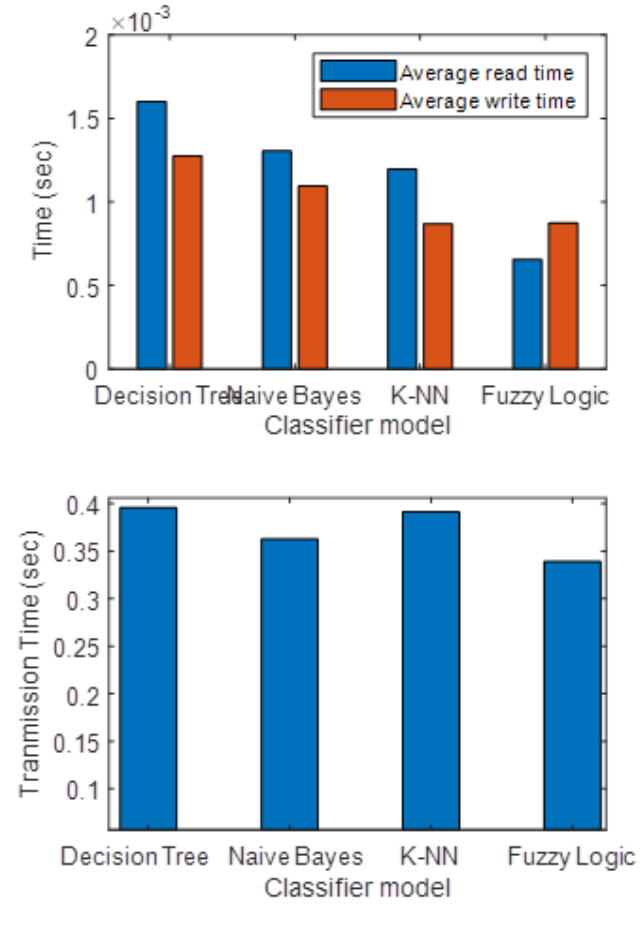

Figure 6: Average read /write time (top) transmission time (bottom)

Highest accuracy obtained for holdout validation cases compared to k-fold validation cases. Fuzzy logic gives highest accuracy value of 69.3% at 45% training data kept at hold out for validation. It shows that the performance of ML models is not simple by default parameters. All ML methods have different domain of parameter set at which it gives best performance. Lowest misclassification cost is also exhibited by fuzzy logic at same hold out value. Decision tree takes low training time but possess very low accuracy hence not suitable for this process. The fuzzy logic at highest accuracy also shows second highest value of observation per second. Performance over the phae of transmission of data is shown in figure 5 as the plots demonstrating performance as (a) Delay (b) Read time (d) delay server to client (e) read time client (f) write time client. The summarized value shown in the figure 6 as the average read /write time (top) transmission time (bottom) during the transmission of data packets between different clients through the common server.

5. Conclusions

In this research, we present a machine learning-based TCP protocol that learns from both end-to-end and intra-network data. Our evaluation, which uses different ML implementation and real-world traces, shows that the proposed scheme performs well across a wide range of network conditions, achieving faster transmissions, lower delays, and packet loss rates than most existing protocols in the majority of tested scenarios. We also examined stability and fairness, demonstrating that it is less aggressive than other protocols when competing with various transport methods and instances of itself from many sources. Furthermore, our findings show that integrating network-layer information considerably improves the performance of the network congestion condition that is accessible at the source. For example, the learning policy's trade-off between exploration and exploitation has an impact on performance. Furthermore, in network conditions that differ greatly from those examined in our evaluation—such as satellite or data center networks—its performance may vary. Despite the positive results, one critical unanswered issue remains: can the machine learning model utilized be effectively ported to different network scenarios? AI and machine learning researchers are actively working on knowledge transfer and machine learning's wider generalizability. Investigating their relevance to congestion control might yield significant insights for adjusting to a broader range of network situations.

Refrences

[1] A. Sacco, F. Matteo, F. Esposito, and G. Marchetto, “Owl: Congestion control with partially invisible networks via reinforcement learning,” in IEEE INFOCOM-IEEE Conference on Computer Communications. IEEE, 2021, pp. 1–10.
[2] V. Paxson, M. Allman, J. Chu, and M. Sargent, “Computing TCP’s retransmission timer,” RFc 2988, November, Tech. Rep., 2000.
[3] J. Huang, F. Qian, Y. Guo, Y. Zhou, Q. Xu, Z. M. Mao, S. Sen, and O. Spatscheck, “An in-depth study of LTE: effect of network protocol and application behavior on performance,” in ACM SIGCOMM Computer Communication Review, vol. 43. ACM, 2013, pp. 363–374.
[4] H. Jiang, Y. Wang, K. Lee, and I. Rhee, “Tackling bufferbloat in 3G/4G networks,” in Proceedings of the 2012 Internet Measurement Conference. ACM, 2012, pp. 329–342.
[5] K. Winstein and H. Balakrishnan, “Tcp ex machina: Computer-generated congestion control,” ACM SIGCOMM Computer Communication Review, vol. 43, no. 4, pp. 123–134, 2013.

[6] K. Winstein, A. Sivaraman, and H. Balakrishnan, "Stochastic forecasts achieve high throughput and low delay over cellular networks," in 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13), 2013, pp. 459–471.
[7] Y. Zaki, T. Potsch, J. Chen, L. Subramanian, and C. G ¨ org, "Adaptive ¨ congestion control for unpredictable cellular networks," in ACM SIGCOMM Computer Communication Review, vol. 45. ACM, 2015, pp. 509–522.
[8] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data Center TCP (DCTCP)," in Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '10), 2010, p. 63–74.
[9] C. Wilson et al., "Better never than late: Meeting deadlines in datacenter networks," in ACM SIGCOMM Computer Communication Review, vol. 41. ACM, 2011, pp. 50–61.
[10] W. Li, F. Zhou, W. Meleis, and K. Chowdhury, "Learning-based and data-driven TCP design for memory-constrained IoT," in 2016 International Conference on Distributed Computing in Sensor Systems (DCOSS). IEEE, 2016, pp. 199–205.
[11] B. V. Ramana, B. Manoj, and C. S. R. Murthy, "Learning-TCP: A novel learning automata based reliable transport protocol for ad hoc wireless networks," in 2nd International Conference on Broadband Networks (BROADNETS). IEEE, 2005, pp. 484–493.
[12] A. Sacco, F. Esposito, and G. Marchetto, "Rope: An architecture for adaptive data-driven routing prediction at the edge," IEEE Transactions on Network and Service Management, vol. 17, no. 2, pp. 986–999, 2020.
[13] V. Badarla and C. S. R. Murthy, "Learning-TCP: A stochastic approach for efficient update in TCP congestion window in ad hoc wireless networks," Journal of Parallel and Distributed Computing, vol. 71, no. 6, pp. 863–878, 2011.
[14] M. Dong, Q. Li, D. Zarchy, P. B. Godfrey, and M. Schapira, "PCC: Re-architecting Congestion Control for Consistent High Performance," in 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15), Oakland, CA, 2015, pp. 395–408.
[15] V. Arun and H. Balakrishnan, "Copa: Practical delay-based congestion control for the internet," in 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18), 2018, pp. 329–342.
[16] Y. Kong, H. Zang, and X. Ma, "Improving tcp congestion control with machine intelligence," in Proceedings of the 2018 Workshop on Network Meets AI & ML, 2018, pp. 60–66.
[17] 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, 2018.
[18] P. Goyal, A. Agarwal, R. Netravali, M. Alizadeh, and H. Balakrishnan, "ABC: A simple explicit congestion controller for wireless networks," in 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20), 2020, pp. 353–372.
[19] S. Floyd, "TCP and explicit congestion notification," ACM SIGCOMM Computer Communication Review, vol. 24, no. 5, pp. 8–23, 1994.
[20] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529–533, 2015.
[21] N. Jay, N. Rotman, B. Godfrey, M. Schapira, and A. Tamar, "A deep reinforcement learning perspective on internet congestion control," in Proceedings of the 36th International Conference on Machine Learning (ICML). PMLR, 2019, pp. 3050–3059.
[22] S. Ha, I. Rhee, and L. Xu, "Cubic: a new tcp-friendly high-speed tcp variant," ACM SIGOPS operating systems review, vol. 42, no. 5, pp. 64–74, 2008.
[23] J. C. Hoe, "Improving the start-up behavior of a congestion control scheme for tcp," ACM SIGCOMM Computer Communication Review, vol. 26, no. 4, pp. 270–280, 1996.
[24] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to end congestion avoidance on a global Internet," IEEE Journal on selected Areas in communications, vol. 13, no. 8, pp. 1465–1480, 1995.
[25] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "Bbr: Congestion-based congestion control," Queue, vol. 14, no. 5, pp. 20–53, 2016.
[26] Geni, Exploring Networks of the Future. Accessed: 2022-10-15. [Online]. Available: https://www.geni.net/
[27] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "A compound tcp approach for high-speed and long distance networks," in IEEE INFOCOM-IEEE Conference on Computer Communications. IEEE, 2006, pp. 1–12.
[28] C. Jin, D. X. Wei, and S. H. Low, "Fast tcp: motivation, architecture, algorithms, performance," in IEEE INFOCOM-IEEE Conference onComputer Communications, vol. 4. IEEE, 2004, pp. 2490–2501.
[29] S. Park, J. Lee, J. Kim, J. Lee, S. Ha, and K. Lee, "Exll: An extremely low-latency congestion control for mobile cellular networks," in Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies (CoNEXT '18), 2018, pp. 307–319.

[30] R. Boutaba et al., "A comprehensive survey on machine learning for networking: Evolution, applications and research opportunities," Journalof Internet Services and Applications, vol. 9, 05 2018.
[31] H. Jiang, Y. Luo, Q. Zhang, M. Yin, and C. Wu, "TCP-Gvegas with prediction and adaptation in multi-hop ad hoc networks," Wireless Networks, vol. 23, no. 5, pp. 1535–1548, 2017.
[32] V. Badarla and C. Siva Ram Murthy, "A novel learning based solution for efficient data transport in heterogeneous wireless networks," Wireless Networks, vol. 16, no. 6, pp. 1777–1798, 2010.
[33] M. Dong, T. Meng, D. Zarchy, E. Arslan, Y. Gilad, B. Godfrey, and M. Schapira, "PCC vivace: Online-learning congestion control," in 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18), 2018, pp. 343–356.
[34] T. Meng, N. R. Schiff, P. B. Godfrey, and M. Schapira, "PCC proteus: Scavenger transport and beyond," in Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '20), 2020, pp. 615–631.
[35] S. Abbasloo, C.-Y. Yen, and H. J. Chao, "Classic meets modern: A pragmatic learning-based congestion control for the internet," in Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '20), 2020, p. 632–647.
[36] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe, "Design and implementation of a routing control platform," in USENIX Symposium on Networked Systems Design and Implementation (NSDI 05), 2005, pp. 15–28.
[37] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," in Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '02). Association for Computing Machinery, 2002, pp. 89–102.
[38] S. Abbasloo, Y. Xu, H. J. Chao, H. Shi, U. C. Kozat, and Y. Ye, "Toward optimal performance with network assisted TCP at mobile edge," in 2nd USENIX Workshop on Hot Topics in Edge Computing (HotEdge), 2019.
[39] Y. Li, R. Miao, H. H. Liu, Y. Zhuang, F. Feng, L. Tang, Z. Cao, M. Zhang, F. Kelly, M. Alizadeh et al., "HPCC: High precision congestion control," in Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM '19). Association for Computing Machinery, 2019, pp. 44–58.
[40] G. Kumar, N. Dukkipati, K. Jang, H. M. G. Wassel, X. Wu, B. Montazeri, Y. Wang, K. Springborn, C. Alfeld, M. Ryan, D. Wetherall, and A. Vahdat, "Swift: Delay is simple and effective for congestion control in the datacenter," in Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '20), 2020, p. 514–528.