Coarse Estimation of Bottleneck Router's Buffer Size for Heterogeneous TCP Sources

Christian Vega Caicedo[1,3], Jorge E. Pezoa[1], Elie F. Kfoury[2], and Jorge Crichigno[2]

[1]Departamento de Ingeniería Eléctrica, Universidad de Concepción, Concepción, Chile
[2]Integrated Information Technology Department, University of South Carolina, USA
[3]Facultad de Ingeniería, Universidad CESMAG, Pasto, Colombia
christianvega@udec.cl, jpezoa@udec.cl, ekfoury@email.sc.edu, jcrichigno@cec.sc.edu

***Abstract*—Although the importance of router's buffer sizing in network performance is well known, estimating the current size of the bottleneck buffer is an open research problem. This paper presents a method to achieve such estimation, for the case where the bottleneck buffer operates under a finite number of buffer sizing regimes. The scheme uses a supervised machine learning approach to properly model such regimes and a classification mechanism to predict the coarse buffer size using the following end-to-end network measurements, which are collected at the sender side: throughput, Round Trip Time (RTT), and Congestion Window (CWND). In contrast to previous work, the scheme does not assume a homogeneous congestion control algorithm used by the senders. The proposed approach was tested using data collected on a real testbed. The corresponding results show that the Support Vector Machine (SVM) Radial Basis Function (RBF) classifier correctly estimates the bottleneck buffer size, under different network conditions.**

***Index Terms*—Buffer Size, Machine Learning, Estimation, Feature Engineering, Congestion Control.**

I. Introduction

Router's buffer sizing in a communication network has been of interest to researchers due to its significant impact on the overall performance in a wide variety of network deployments. If buffers are too small, the network is prone to experience a high packet loss rate. On the other hand, if buffers are too large, the latency and complexity as well as the router's cost increase [1]. This situation has motivated studies and models for determining the appropriate amount of buffer size that should be configured. A particularly important device is the router's interface that constitutes the bottleneck link, because it is where persistent queues form. Additionally, knowing the queue occupancy is beneficial for decisions of Congestion Control (CC) and Active Queue Management (AQM) schemes [2].

While broad studies have been conducted in the past, the advent of new congestion control algorithms that do not adhere to the additive increase multiplicative decrease (AIMD) control law requires new examinations [3]. In recent work, Abbasloo *et al.* addressed the bottleneck link state as an estimation problem obtaining models from traffic patterns [4]. A limitation of this work is the assumption of a homogeneous congestion control algorithm, which contrasts with actual traffic characterized by the use of multiple congestion control algorithms such as CUBIC [5], Bottleneck Bandwidth and Round-trip (BBR) [6], and BBR version 2 (BBRv2) [7] algorithms.

End-to-end measurements are obtained by the continuous exchange of information between the sender and the receiver in a TCP connection. This collected information describes patterns about the state of the network that can be exploited to estimate its parameters. The advantage of the end-to-end approach is that it does not generate overhead on the network and does not require intervention in forwarding devices' configuration. Due to phenomena associated with congestion such as packet loss, retransmissions and queuing delays, and the presence of heterogeneous flows, estimating the bottleneck buffer size from end-to-end measurements is not a trivial task. Therefore, the bottleneck buffer size cannot be obtained from classical models or simple rules. In this way, Machine Learning (ML) algorithms emerge as an alternative to deal with complex estimation problems in networks from the interaction between traffic patterns.

In this paper, we aim to answer the following research question: Is it possible to infer, at the sender side, the buffer size of a bottleneck link using the summarized information collected from network measurements and processed with state-of-the-art ML algorithms? We focused on yielding a coarse estimate for the current buffer size configured at the network bottleneck to answer this question. Thus, we assumed that the current buffer size takes a finite set of values, expressed in Bandwidth Delay Product (BDP) units. Further, we removed the assumption that end nodes use a homogeneous CC algorithm. We carried out extensive measurements on a real testbed to collect data. Using ML, we identified that average throughput, Round Trip Time (RTT), and Congestion Window (CWND) metrics, collected at the sender side, are convenient features to determine the buffer size at the bottleneck link interface. Our results show that the buffer size estimation is correct, even when network conditions slightly differ from the training data. Applications of our solution include configuring the router's buffer size, selecting the congestion control algorithm at end devices, dynamically modifying the TCP sending rate using pacing [8], and supporting learning-based CC algorithms. The proposed scheme is suitable also for high-throughput high-latency networks used to transfer big data, such as Science Demilitarized Zones (DMZs) [9].

This work has been supported by ANID: FONDECYT Regular 2020 Folio 1201495, PCHA/Doctorado Nacional Folio 2018-21180418.

II. Related Work

There exists a considerable body of literature around buffer sizing. Several studies have shown that the network performance is affected by the buffer size setting in routers. In earlier works, the buffer size was configured considering the link utilization. In [10], the pervasive BDP rule-of-thumb for buffer sizing is defined. Such a rule states that the buffer size must be equal to the product between the link's capacity, C, and the RTT. This buffer size guarantees full utilization of the link. The results were derived by observing the behavior of a small number of simultaneous TCP flows. This rule was widely used by Internet service providers (ISPs) and manufacturers when configuring and designing routers. However, in practice, the number of flows sharing a link is usually large. Appenzeller et al. [11] incorporated this observation and proved that for N desynchronized flows, the bottleneck link could be kept fully utilized with a buffer size of approximately $C \cdot RTT/\sqrt{N}$. Note that this new rule assumes that flows use long-lived TCP New Reno congestion control [12], which is based on the additive increase multiplicative decrease (AIMD) behavior. Thus, it does not apply to new congestion control algorithms based on pacing, such as BBR [6] and BBRv2 [7]. Supported by the pacing technique, authors in [13] hypothesize that it is possible to obtain a remarkable network performance by further reducing the buffer size to a few dozen packets. This approach can be useful when the network administrator has full control of network resources.

Previous works showed that both the congestion control protocol and the buffer size play a crucial role in the network's performance because they are closely related. Analyses of delay-based congestion control protocols suggested that the performance of Transmission Control Protocol (TCP)-Vegas and TCP-FAST can be susceptible to the precise choice of small buffers [14]. In [4], the authors tested Orca, their own Learning-Based Congestion Control, for buffer sizes in the range of 3KB to 96MB. Results reported that Orca outperforms CUBIC [5] under any buffer size condition. Recently, Kfoury et al. [7] conducted tests for different buffer sizes, for CUBIC, BBR, BBRv2, and TCP variants. They conclude that BBR and BBRv2 are suitable with a small buffer size in terms of throughput and link utilization.

Experiments on buffer sizes have not been limited to controlled simulation of laboratory testbeds. In [15], Facebook's engineering team conducted a series of experiments on its backbone using real network traffic to measure the impact of buffer size reduction. The effect of buffer sizing on the Quality of Experience (QoE) metrics was measured at Netflix's network [16]. Researchers collected data using Netflix Open Connect Architecture and production traffic from users during peak hours. They observed that video QoE could degrade when both too small and too large buffer sizes are configured. The authors also state that more experiments are needed to deeply understand buffer size and CC in video traffic performance.

When the bottleneck link parameters are not known or updated to the senders, the estimation of buffer parameters can support decision making in congestion protocols. The Delay Friendly TCP (DFTCP) [17] performs rate control based on the estimation of the maximum size of available buffers in the routers along a network flow path. The model computes this value from average RTT and propagation delay. The parameter estimation also can be carried out from network measurements. Consequently, it is necessary to collect meaningful information from the network in a training process aiming to abstract the network's operation patterns in a general way. Authors in [18] added a bottleneck link capacity estimator to BBR to improve the performance of multipath TCP connections. Ciaccia *et al.* [19] proposed a statistical method based on traffic patterns, especially at the slow-start phase, to estimate the bottleneck's link capacity using passive measurements. The authors also train a neural network to improve the estimation accuracy of the model. ML techniques allow finding hidden patterns that favor bottleneck parameters estimation in noisy and bursty cross-traffic, multiple bottleneck links, and inaccurate time stamping. This approach is exploited in [20] where authors estimate the available bandwidth using a reinforcement learning model. The buffer size also impacts metrics such as link utilization, throughput, and resource consumption. Optimizing such metrics simultaneously was studied in [21].

III. Materials and Methods

To carry out the buffer size estimation, we used the workflow based on the supervised ML approach depicted in Fig. 1. In brief, the methodology is divided into four stages, namely: data collection, feature engineering, training, and evaluation.

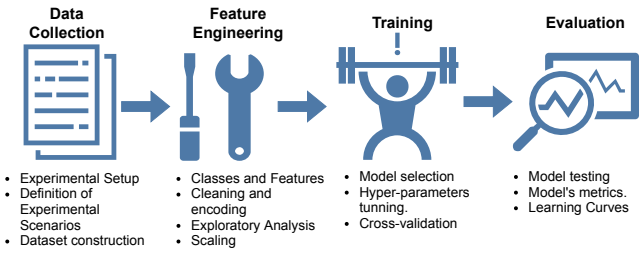


Fig. 1. A workflow for the coarse buffer size estimation presented in this paper.

A. Data Collection

1) Experimental Setup: Figure 2 shows the topology used to collect data. The testbed is an extension of the well-known dumbbell topology. The working scenario consists of 100 sender nodes, labeled as h1 to h100, and 100 receiver nodes, labeled as h101 to h200. For simplicity, sender and receiver nodes were emulated on isolated namespaces using Mininet and deployed in two separate servers to ensure sufficient computational resources. During the experiments, each sender node establishes a single TCP connection with one receiver node. TCP Cubic, BBR, and BBRv2 congestion control algorithms were configured at the sender nodes to introduce heterogeneity in the way senders react and consequently in the

measurements. The Netem tool was used at the switch labeled as S1 to configure different propagation delays and random packet loss rates. The propagation delay parameter was set to 20 ms, while packet loss percentages were set to 0%, 1%, 2%, and 3%. We established the bottleneck link rate in 1Gbps at the interface linking routers labeled as R1 and R2. Lastly, we qualitatively describe the bottleneck link-state coarsely using a finite set of values for the buffer size, expressed in BDP units. Considering the above setup, BDP is the product of 1Gbps and 20ms (i.e., 2.5×10^6 bytes). Thus, the buffer size at R1 was configured at 0.01 BDP, 0.1BDP, 1BDP, 10BDP, and 100BDP representing a very small, a small, a moderate, a large, and a very large buffer sizing regime. In all regimes, the buffer size is greater than the expected size of a TCP segment.

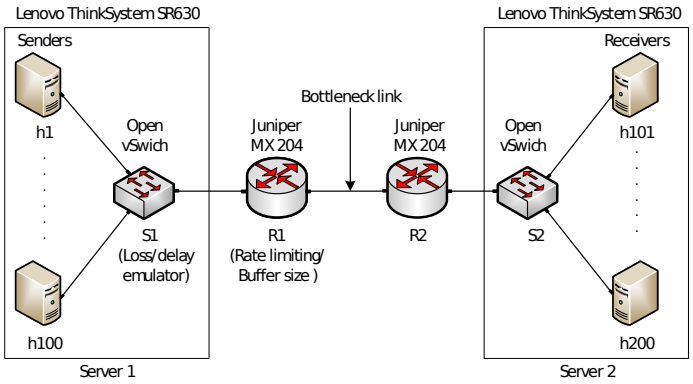


Fig. 2. Topology used for data collection.

2) Definition of experimental scenarios: Here, an experimental scenario represents a network condition, given by a packet loss level and buffer a size regime, evaluated by exchanging information between the sender and receiver nodes. Table I lists the settings for the experimental scenarios evaluated in the study. We used a nearly even distribution of congestion control algorithms assigned to hosts and a no packet loss scenario for ML training and validation purposes. We employed an unbalanced distribution of such congestion control algorithms to test our coarse estimation method for the bottleneck router buffer size. We also used four different packet loss levels during testing to assess whether our inference method can generalize its predictions under unseen training scenarios. In total, five scenarios were defined for training and validation, and 20 scenarios for testing. For each scenario, we collected measurements of throughput, RTT and CWND using iPerf3. Each scenario was executed for 120 seconds.

3) Construction of the ML Dataset: Measurement traces were processed to compute, at each host and for every scenario, the following time-averaged metrics: throughput, RTT, and CWND at the sender side. Also, we labeled the data with its corresponding CC protocol and buffer size regime. The training and validation dataset was built by joining the data provided by the scenarios with no packet loss and the five defined buffer size regimes in a balanced distribution of host CC algorithms. The remaining scenarios with unbalanced CC protocol distribution were used to build four test datasets, one for each packet loss level.

TABLE I
EXPERIMENTAL SCENARIOS

Phase	Host's CC Algorithm	Packet Loss (%)	Buffer Size (BDP)
Training and validation	34 CUBIC 33 BBR 33 BBR2	0	0.01, 0.1, 1, 10, 100
Test	60 CUBIC 25 BBR 15 BBR2	0, 1, 2, 3	0.01, 0.1, 1, 10, 100

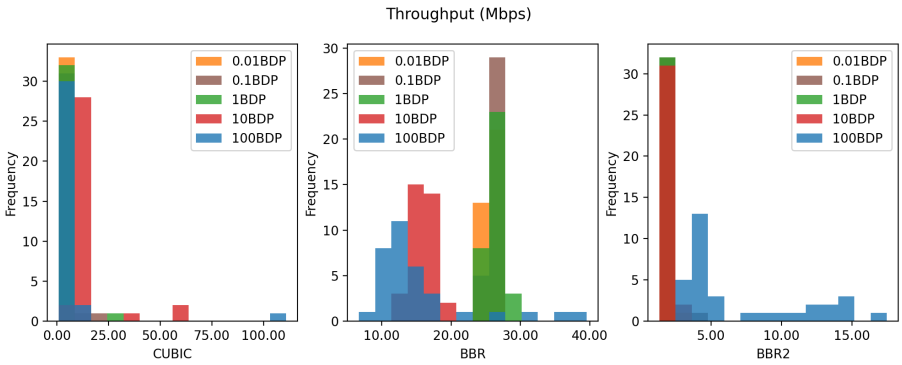
B. Feature Engineering for the Coarse Buffer Sizing Regimes

1) On the Classes and Features: Using an ML approach, we define a representation space for the coarse buffer sizing regime in terms of explanatory variables termed as features. From the networking literature, relevant feature variables for describing end-to-end connections are the average throughput (in Mbps), the RTT (in ms), the CWND (in MB), and the congestion protocol. The average throughput, the average RTT, and the CWND features are real-valued, while the congestion protocol feature is categorical. Such end-to-end features are used, in turn, as proxies for the latent space representing the coarse buffer sizing regimes at the bottleneck router. Thus, we can state the problem of inferring the buffer sizing regime as a multi-class discrimination problem where each class is mapped onto a single coarse buffer size configured at the router.

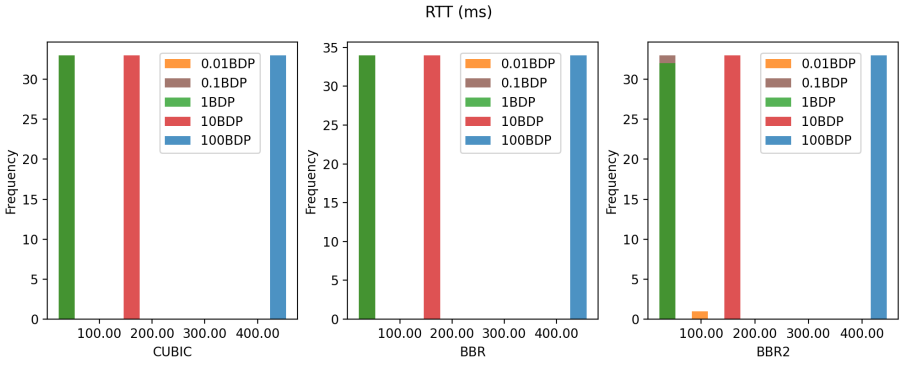
2) Dataset cleaning and Enconding: Firstly rows with missing values due to data corruption or record data failures were deleted from the dataset to yield robust classification models. In addition, the categorical values associated with the congestion control protocols and buffer size regimes were encoded using integer values to match the requirements of the classification methods used in the present study.

3) Exploratory Analysis of Feature Variables: Table II lists the descriptive statistics for the average throughput, the RTT, and the CWND feature variables in the training ad validation dataset. Features exhibit a large variance, as evidenced by both the extreme values and the coefficient of variation. We also estimated the empirical probability distributions for each feature by computing their histograms. In such calculations, we used the Freedman-Diaconis rule for binning. Histograms in Fig. 3 show that classes (buffer sizing regimes) overlap at several intervals in the feature space. Therefore, it can be noted that variables cannot be used independently to infer the coarse buffer size at the bottleneck router. Figure 4 shows the correlation matrix between the features variables. It can be noted that features do not exhibit a high degree of correlation, meaning that no feature variable should be discarded during training.

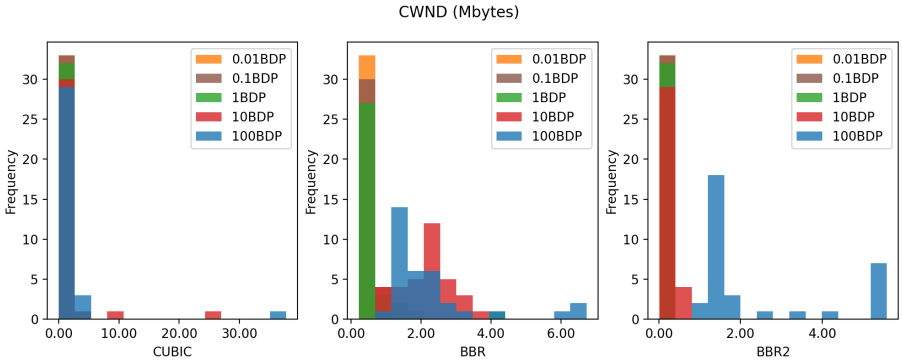
4) Scaling: Since data ranges for the feature variables are very different, we normalized the non-categorical variables in the range [0,1] to overcome the units' differences and contribute to the model generalization.



(a)



(b)



(c)

Fig. 3. Histogram of the non-categorical feature variables: (a) Average end-to-end throughput. (b) Average RTT. (c) Average CWND.

TABLE II
DESCRIPTIVE STATISTICS OF THE TRAINING AND VALIDATION DATASETS FOR THE NON-CATEGORICAL FEATURE VARIABLES.

Stat	Throughput (Mbps)	RTT (ms)	CWND (MB)
Mean	10.419	138.954	0.906
Standard Deviation	11.220	162.415	2.341
Min value	0.800	21.802	0.007
Max value	110.976	457.592	37.732
Coefficient of variation	1.077	1.17	2.583

C. Model Training

Six classic and state-of-the-art classifiers were trained to infer the coarse buffer sizing regimes. The trained classifiers are the Least Squares (LS), the Gaussian Naive Bayes (GNB), the Linear-kernel Support Vector Machine (SVM), the Radial Basis Function (RBF) kernel SVM, the K-Nearest Neighbors (KNN), and the Multi-Layer Perceptron Classifier (MLPC). Here, we use the LS classifier as a baseline instead of using the typical Zero Rule or random picking classifier.

We used a stratified shuffle split approach for cross-validation with five iterations and a test-size of 20%. To tune the hyperparameters associated with each classifier, we carried out an exhaustive search over a grid of candidate parameters. Classification Accuracy (Acc) metric was chosen as the primary performance measure. Acc is defined as the ratio of correct predictions to the total number of input dataset

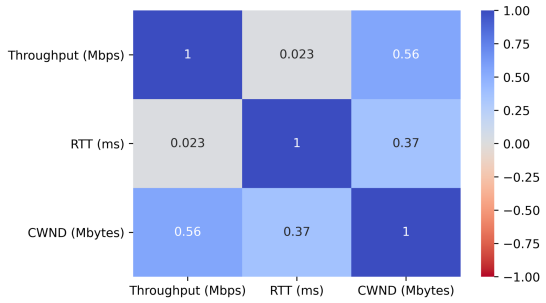


Fig. 4. Correlation matrix for the non-categorical feature variables.

instances. A correct prediction means that the algorithm is able to determine the state of the network in terms of bottleneck router's buffer size from end-to-end measurements. Table III lists both the parameters and values used to train the classifiers. Training and validation were carried out in Python, exploiting the benefits of the Scikit-learn library [22]. In summary, hyperparameter tuning was employed to achieve better classification results, while cross-validation was used to avoid overfitting and favor the models' generalization.

TABLE III
TRAINING VALUES FOR THE HYPERPARAMETERS AT EACH CLASSIFIER.

Classifier	Parameters	Range	Spacing	Values
SVM-Lineal	C	$[1, 10^5]$	Log	6
SVM-RBF	C	$[10^{-3}, 10^{10}]$	Log	20
	gamma	$[10^{-9}, 10^3]$	Log	20
KNN	n_neighbords	[2,9]	Linear	8
MLPC	max_iter	[1000, 4000]	Linear	4
	alpha	$[10^{-1}, 10^{-4}]$	Linear	4
	hidden_layers	[1,14]	Linear	14
	random_state	[1,9]	Linear	9

Next, we present the results that support our claim that we can infer, at the sender side, the coarse buffer size of a bottleneck link, using the summarized information collected from network measurements and processed with state-of-the-art ML algorithms.

IV. RESULTS AND EVALUATION

Table IV summarizes the performance of classifiers designed in this paper at the model validation stage in terms of Acc. We provide the best values for the trained models' hyperparameters, that is, those with which the maximum ACC was obtained.

We comment first that our problem's baseline classifier performance is $Acc = 0.45$. (We note that for a five-class Zero Rule detector, its performance is 0.2.). Out of the five trained classifiers presented here, the lowest performance at the validation stage was obtained by LS classifier. Such classifier achieved a classification accuracy of 0.45. This is attributed to its inability to separate the classes in the feature space domain. The highest performance was obtained by the RBF-based SVM classifier, whose classification accuracy of very close to 1 during validations.

Figure 5 compares the learning curves of proposed classifiers during the training and validation stages as a function of the number of examples provided to the model. In all classifiers, we obtain a minimal gap (i.e. less than 0.05 of Acc) between training and test curve. This finding indicates that models deal correctly with underfitting and overfitting issues and also that the training dataset was representative to provide sufficient information for model construction. It is observed that as data are added to the linear SVM and RBF training set, they keep a remarkable and constant performance, and these need a reduced number of instances to reach a suitable performance. We also note that GNB-based and SVM lineal classifiers reduce its performance as the number of training samples increases. This behavior is attributed to the inaccurate assumption about the feature variables being jointly independent, as stated depicted in Fig. 4.

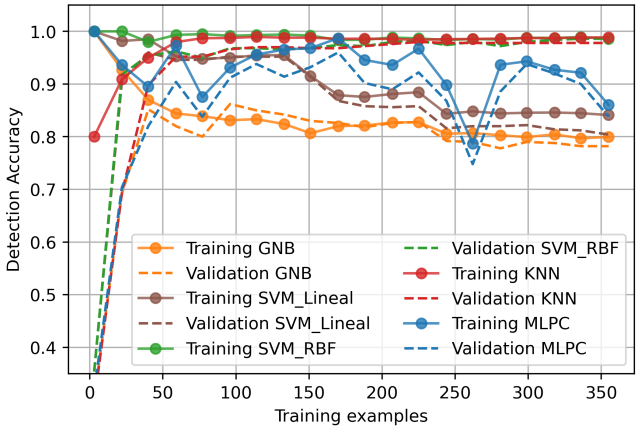


Fig. 5. Learning curves of the classifiers.

To assess our inference models' ability to generalize their results, we inferred the coarse buffer sizes using test data not used during the training and validation stages. In the first test scenario, which considers imbalanced CC algorithms and no packet loss conditions, the classifiers keep similar performances to those presented during the validation stage. These results show that the feature variables used here are suitable to describe a particular buffer size condition. In the remaining three test scenarios, which also consider imbalanced CC algorithms yet introduce different levels of packet losses, we observe again that the classifiers' performance decreases in small fractions. We highlight that the linear-kernel SVM and the MLPC model generalize their results fairly well, since their classification accuracy decreases less than 3% concerning the no pack loss test scenario. Above this level of packet loss, the accuracy of the models degrades significantly. Although a packet loss level greater than 3% is uncommon in our network scenario, this is a situation that can occur, for example, if wireless links are present in the access network. Therefore future works can be conducted in that direction.

TABLE IV
SUMMARY OF THE CLASSIFICATION RESULTS FOR ALL THE COARSE BUFFER SIZING ESTIMATORS.

Classifier	Validation			Test			
	Acc	Parameter	Best	Acc No Loss	Acc Loss 1%	Acc Loss 2%	Acc Loss 3%
LS	0.45			0.45	0.40	0.40	0.40
Gaussian NB	0.8			0.98	0.80	0.77	0.75
SVM-Linear	0.86	C	10	0.89	0.99	0.97	0.89
SVM-RBF	0.99	C	10^{10}	0.99	1.0	0.97	0.92
		gamma	2.63×10^{-5}				
KNN	0.98	n_neighbords	2	0.98	0.98	0.96	0.91
MLPC	0.98	max_iter	3000	0.93	0.95	0.93	0.90
		alpha	0.1				
		hidden_layers	13				
		random_state	4				

V. CONCLUSION

This paper provides a suitable method to coarsely estimate the buffer size regime at the bottleneck link, following the supervised ML approach. Our findings indicate that average values of throughput, RTT, and CWND and the CC algorithm used by the sender-side are useful features for building buffer size inference models. Out of our six trained models, the RBF-based SVM classifier outperforms the remaining ones in estimating the coarse buffer size. This finding is attributed to its ability to adapt to the classes' non-linear separability evidenced in the exploratory analysis. Furthermore, this classifier exhibited a robust performance in the presence of heterogeneous unbalanced sources when packet losses are induced in the network. Our approach's most promising attribute is that it might be used to support congestion control strategies based on learning models or explicit feedback notifications. As future work, we will aim to integrate our ML estimation models in the on-line operation of CC algorithms, with measurements and actions supported in data-plane programmable devices for Science DMZ cyberinfrastructures.

ACKNOWLEDGMENT

We thank J. Gomez for his support with network tests conducted at the University of South Carolina.

REFERENCES

[1] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing Router Buffers," in *Proc. Conf. Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 281–292.

[2] S. Arslan and N. McKeown, "Switches Know the Exact Amount of Congestion," in *Proceedings of the 2019 Workshop on Buffer Sizing*. New York, NY, USA: Association for Computing Machinery, 2019.

[3] N. McKeown, G. Appenzeller, and I. Keslassy, "Sizing router buffers (redux)," *SIGCOMM Comput. Commun. Rev.*, vol. 49, no. 5, p. 69–74, Nov. 2019.

[4] S. Abbasloo, C.-Y. Yen, and H. J. Chao, "Classic Meets Modern: A Pragmatic Learning-Based Congestion Control for the Internet," in *Proc. Conf. ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. SIGCOMM '20. New York, NY, USA: ACM, 2020, p. 632–647.

[5] 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.

[6] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: congestion-based congestion control," *Communications of the ACM*, vol. 60, no. 2, pp. 58–66, 2017.

[7] E. F. Kfoury, J. Gomez, J. Crichigno, and E. Bou-Harb, "An emulation-based evaluation of TCP BBRv2 Alpha for wired broadband," *Computer Communications*, vol. 161, pp. 212 – 224, 2020.

[8] E. F. Kfoury, J. Crichigno, E. Bou-Harb, D. Khoury, and G. Srivastava, "Enabling TCP pacing using programmable data plane switches," in *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*. IEEE, 2019, pp. 273–277.

[9] J. Crichigno, E. Bou-Harb, and N. Ghani, "A comprehensive tutorial on science DMZ," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 2041–2078, 2019.

[10] C. Villamizar and C. Song, "High Performance TCP in ANSNET," *SIGCOMM Comput. Commun. Rev.*, vol. 24, no. 5, p. 45–60, Oct. 1994.

[11] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing Router Buffers," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 4, p. 281–292, 2004.

[12] A. Vishwanath, V. Sivaraman, and M. Thottan, "Perspectives on Router Buffer Sizing: Recent Results and Open Problems," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 2, p. 34–39, Mar. 2009.

[13] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden, "Part III: Routers with Very Small Buffers," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 3, p. 83–90, Jul. 2005.

[14] P. Raja and G. Raina, "Delay Based Feedback, Transport Protocols and Small Buffers," in *Proceedings of the Nineteenth International Workshop on Quality of Service*, ser. IWQoS '11. IEEE Press, 2011.

[15] N. Beheshti, P. Lapukhov, and Y. Ganjali, "Buffer Sizing Experiments at Facebook," in *Proc. Workshop Buffer Sizing*, ser. BS '19. New York, NY, USA: ACM, 2019.

[16] B. Spang, B. Walsh, T.-Y. Huang, T. Rusnock, J. Lawrence, and N. McKeown, "Buffer Sizing and Video QoE Measurements at Netflix," in *Proceedings of the 2019 Workshop on Buffer Sizing*, ser. BS '19. New York, NY, USA: Association for Computing Machinery, 2019.

[17] X. Zhang, N. Gu, J. Su, and K. Ren, "DFTCP: A TCP-friendly delay-based high-speed TCP variant," in *2016 International Conference on Networking and Network Applications (NaNA)*, 2016, pp. 273–278.

[18] T. Zhu, X. Qin, L. Chen, X. Chen, and G. Wei, "wBBR: A Bottleneck Estimation-Based Congestion Control for Multipath TCP," in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, 2018, pp. 1–5.

[19] F. Ciaccia, I. Romero, O. Arcas-Abella, D. Montero, R. Serral-Gracià, and M. Nemirovsky, "SABES: Statistical Available Bandwidth EStimation from passive TCP measurements," in *2020 IFIP Networking Conference (Networking)*, 2020, pp. 743–748.

[20] S. K. Khangura and S. Akın, "Measurement-Based Online Available Bandwidth Estimation Employing Reinforcement Learning," in *2019 31st International Teletraffic Congress (ITC 31)*, 2019, pp. 95–103.

[21] J. Crichigno, N. Ghani, J. Khoury, W. Shu, and M.-Y. Wu, "Dynamic routing optimization in WDM networks," in *2010 IEEE Global Telecommunications Conference (GLOBECOM)*, 2010.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.