Harnessing ML For Network Protocol Assessment: A Congestion Control Use Case

Parsa Pazhooheshy
University of Toronto
parsap@cs.toronto.edu

Soheil Abbasloo
University of Toronto
abbasloo@cs.toronto.edu

Yashar Ganjali
University of Toronto
yganjali@cs.toronto.edu

ABSTRACT

In this paper, our primary objective is to showcase that the application of machine learning techniques extends beyond network protocol design. We aim to demonstrate that performance assessment of network protocols, a vital aspect of improving network infrastructures and developing better protocol designs, can be modernized through the utilization of machine learning. As a step towards this goal, we have designed and introduced Mahak, the first tool that harnesses active learning techniques to automate the performance assessment of congestion control schemes. Mahak actively learns to optimize the evaluation process of congestion control schemes so that they can generate their performance maps over a desired space without exhaustively testing them in every scenario. Mahak treats schemes under the test as black boxes. This protocol-agnostic aspect of Mahak enables users to directly assess the performance of the actual implementation of a protocol instead of their over-simplified mathematical models or simplified simulated versions.

CCS CONCEPTS

• **Networks** → **Network performance modeling**; **Network performance analysis**; **Transport protocols**;

KEYWORDS

Protocol Assessment, Active Learning, Congestion Control

ACM Reference Format:
Parsa Pazhooheshy, Soheil Abbasloo, and Yashar Ganjali. 2023. Harnessing ML For Network Protocol Assessment: A Congestion Control Use Case. In *The 22nd ACM Workshop on Hot Topics in Networks (HotNets '23), November 28–29, 2023, Cambridge, MA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3626111.3628182

HotNets '23, Nov., 2023, Cambridge, MA, USA

ACM ISBN 979-8-4007-0415-4/23/11…$15.00
https://doi.org/10.1145/3626111.3628182

1 INTRODUCTION

Setting the Context: Assessing the performance of network protocols is crucial in the field of computer networks as it allows for the evaluation of their effectiveness, aids informed decision-making for network operators, and drives advancements in network and protocol design. By comparing different protocol designs and observing their performance across a wide range of scenarios, valuable insights can be gained, enabling the identification of optimal solutions. This process empowers protocol fine-tuning, performance enhancement, and the resolution of bottlenecks or vulnerabilities. Furthermore, observing protocol performance in diverse conditions helps understand their real-world behavior and ensures reliability and scalability.

1.1 Motivations

1. Covering Large Space Is Daunting & Time-Consuming: Achieving a comprehensive assessment of network protocols requires considering diverse conditions and scenarios. The traditional approach involves manually sweeping through a space of parameters and network conditions and evaluating protocols for each setting. However, manually conducting this assessment even for a single protocol is challenging, time-consuming, and can depend on the evaluators' expertise and experience. Considering multiple performance metrics, network configurations, and workload scenarios further increases the complexity of the assessment task.

2. Simplified Representations Are not Enough: One general approach to reducing the complexity of assessing network protocols is through the use of simulations. Simulations provide a controlled environment where protocols can be evaluated under various network conditions and scenarios. However, a limitation of simulation-based assessments is the need for simplifications and abstractions in representing the protocols themselves. These simplifications can lead to a gap between the simulated behavior and the real-world performance of the protocols. The assessment may overlook certain intricacies and nuances of the actual protocol implementation, which are critical in understanding its true behavior and performance. Moreover, even with simulations, the issue with the vast evaluation space remains. Therefore, simulations, while valuable for initial evaluations, are not sufficient in themselves.

These challenges motivated us to rethink a key question: *Can we efficiently and effectively assess the performance of protocols/designs without exhaustively testing them over the entire space or replacing them with simplified versions (compromising accuracy)?*

1.2 The Scope of Our Work & Contributions

To answer this key question, in this paper, we argue that the network protocol and design assessment are required to be modernized. Toward that end, we propose a novel automated approach harnessing advanced machine learning techniques and in particular, for the first time, we utilize *active learning (AL)* in a networking context.

Internet Congestion Control: To that end, in this position paper, we focus on an important family of network protocols, Internet congestion control designs, and utilize it as a use case to elaborate on our approach. The choice of Internet congestion control as a use case comes from the fact that despite being an old research area, it has remained one of the most active and hot research topics in the network community attracting various designs during the last four decades (e.g., [3–7, 9–13, 15–17, 19, 21–23, 25, 27, 29, 32, 38, 39]). In particular, we have observed different CC designs proposed to address congestion in certain scenarios, but after a few years of either using them or experimenting with them, in the end, their limitations and problems have raised to the surface prompting the search for better ones [2, 39]. Thus, a learning-based automated assessment approach can greatly reduce these long design-assessment cycles, leading to the development of more robust and high-performing protocols.

Mahak: We have designed and developed Mahak, a proof-of-concept tool that embodies our approach and leverages active learning techniques for assessing the performance of CC schemes. The main objective of Mahak is to uncover the performance of a given CC scheme across a wide range of scenarios without exhaustively testing every scenario. Instead, Mahak automatically learns which scenarios to examine so that it can effectively capture the scheme's performance across the entire space. In essence, Mahak dynamically learns to adapt its test conditions to optimize the evaluation process for a given CC scheme. However, the realization of such a tool required addressing several practical challenges and design complexities such as: (1) the protocol-agnostic requirement of the design which enforces treating the CC protocol under the test as a black box, (2) the high dimensionality of the test space, and (3) the restriction of utilizing only a small amount of data to make an accurate map of the performance. Addressing these challenges, we make the following contributions in this work:

- We demonstrate that the application of ML techniques can extend beyond the network protocol design (the current main focus in the community) and show that ML can be harnessed to automate the assessment process of the network protocol designs.
- To the best of our knowledge, Mahak is the first solution to employ active learning in a networking context.
- Mahak can make the CC assessment process 25× faster (Section 5.1). Utilizing Mahak, we obtained a performance map of a few CC schemes leading to uncovering some performance issues of them (Section 5.3).
- Mahak's code [1] is publicly available to help and facilitate the development of more advanced active learning-based tools and solutions.

2 RELATED WORK

There are different existing approaches to evaluate protocols. Here, we briefly overview and compare them with Mahak.

Mathematical Modeling: Some use mathematical expressions to understand protocol behavior, strengths, and weaknesses. For example, a previous study [37] focuses on mathematically modeling BBR to understand why it exhibits unfairness when BBR flows compete with loss-based schemes. However, the approach of mathematically modeling protocols is challenging (and usually requires over-simplifications) even for a single particular heuristic scheme and cannot be generalized to other ones. For example, deriving mathematical expressions for many recently proposed CC protocols that employ deep reinforcement learning techniques (e.g., [6]) remains an open problem.

Formal Verification: Schemes in this category aim to prove specific properties of protocols or provide counterexamples (e.g., [8]). However, there are two main issues with this class. First, presenting a counterexample alone is not sufficient for evaluating the performance of a target protocol across an entire space. Second, schemes in this class require a prior mathematical expression of the protocol. For example, before using CCAC [8], users are required to express the target CC algorithm as a first-order logic formula. As mentioned before, formulating a simple heuristic scheme can be difficult, let alone formulating the actual final implementation of more complicated protocols.

Stress Testing: Another category of works argues that by subjecting protocols to stress tests, their flaws and limitations can be identified. Designs in this group depend on effective search algorithms to generate a scenario that reveals the poor performance of the target protocol (e.g., CC-Fuzz [30] employs a genetic search algorithm). However, counterexamples alone are inadequate for evaluating the overall performance of a protocol over an entire region. Moreover, search algorithms (such as genetic algorithms) are susceptible to data noise, necessitating simulations for both the network and the target protocol to ensure identical outcomes across multiple runs, limiting their scope of applicability.

Our work: Considering this landscape, Mahak stands out due to two factors: (1) it is protocol-agnostic, *i.e.,* it operates without modeling the underlying protocol, making it applicable even to assessing complicated protocols such as the ones using complex machine learning techniques and enables directly assessing the final real-world implementation of protocols without relying on simplified assumptions, and (2) it aims to reveal the protocol's performance across the entire space, going beyond bug identification and exhaustive testing, and without the need for experts identifying parts of the space that are potentially important.

3 ACTIVE LEARNING OVERVIEW

In recent years, Deep Learning (DL) has revolutionized various fields, ranging from computer vision and natural language processing to speech recognition and autonomous systems. DL models, with their ability to learn complex non-linear patterns and representations from vast amounts of data, have achieved remarkable performance in various tasks [18], [35] [31] [20]. However, one of the critical challenges in training these models is the need for large amounts of labeled data, which can be costly and time-consuming to obtain.

Considering this landscape, active learning (AL) has emerged as a promising approach to address the challenge of data scarcity and efficient data annotation[34]. AL is a subset selection strategy that actively selects the most informative samples from a large unlabeled dataset and requests annotations from human experts. By iteratively selecting samples that are expected to improve the model's performance the most, AL aims to optimize the use of labeled data, reducing the overall annotation effort and cost. The importance of AL lies in its ability to maximize the learning efficiency of models. Rather than randomly selecting samples for annotation or relying solely on fully labeled datasets, AL enables the models to learn from a carefully curated subset of labeled examples. In other words, the model not only can be boosted by training over data but also encouraged to be *curious* about the data that is trained on and actively attempt to choose better examples. This targeted approach allows for more focused and effective model training, leading to improved performance. Fig. 1 depicts the big picture of AL. In a nutshell, first, specific data points from U, the set of all unlabeled data, are selected based on an active query selection algorithm. Then, labels of the

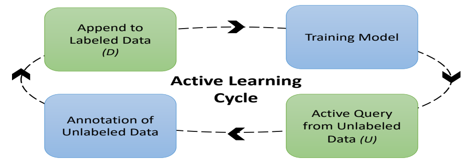

Figure 1: Active Learning Loop

chosen data points are queried from the oracle. In the next step, the newly labeled data points are added to D, the pool of labeled data points. Then, D is fed into the learning block to improve the model. The cycle continues till either the system reaches a predefined labeling cost/budget (a limitation on D) or a predefined target accuracy for the model.

4 MAHAK: DESIGN AND COMPONENTS

Transforming Assessment Task to AL: The key to satisfying our primary goal (uncovering the performance of a scheme across a wide range of scenarios without exhaustively testing it in every scenario) is to generate an accurate *performance model* for the given protocol/design. With this model, we can simply input the identifying factors of a scenario and obtain the scheme's performance without actually evaluating it over that specific scenario. Viewing the assessment process from this perspective and considering that <evaluating a scheme in a scenario to know its performance> is just another form of <querying an oracle to label data> (as it appears in AL terminology), it becomes evident why active learning (AL) is a natural fit for our context.

Mahak utilizes AL to learn a supervised performance model while operating within a limited budget for labeling data (read it as imposing a limitation on the number of evaluated scenarios). Considering the AL terminology (see Section 3), the unlabeled dataset U, is the set containing every possible combination of network space parameters determined by the user as space boundaries. On the other hand, the labeled dataset, D, is a set containing the mentioned input features alongside the performance of the target CC scheme (the label). That formulates D as $D = \{u_i, f(u_i)\}\ for\ u_i \in U$. Mahak starts with an empty set of D, adds values to D in each cycle, and terminates when $|D|$ reaches a given computation budget. Fig. 2 shows the big picture of Mahak.

4.1 Mahak and Processing Inputs/Output

Output: Mahak's output is a mapping from every point in the space (determined by input features) to the estimated performance of the scheme on that point.

Input Features: Mahak takes three sets of inputs from the user: (1) space dimensions and boundaries, (2) the performance metric, (3) the computation budget. Several features

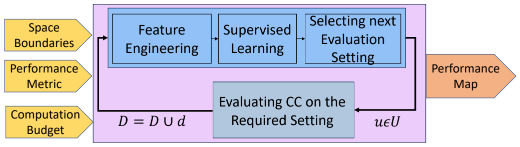

Figure 2: Mahak's Block Diagram

such as available throughput, minimum intrinsic round-trip delay (mRTT), number of competing flows, stochastic loss, changes in link bandwidth, etc. can impact a CC scheme's performance, thus determining different dimensions of the evaluation landscape. In this proof-of-concept version, we consider link capacity, mRTT (minimum round trip time), buffer size, and bandwidth change ratio as the space input features and their minimum and maximum values determining space boundaries. In other words, for each point $u \in U$, we have: $u \in \mathbb{R}^{+4}$. Mahak's second input is the performance metric. The performance metric can be linear (e.g., link utilization) or non-linear (e.g., $-log(delay)$). The last input is the computation budget which governs the limitations on the number of actual evaluations (oracle queries).

Feature Processing: The first step in Mahak's feature processing block is to create multiple different linear/non-linear features out of given input ones. Creating these features can help and expedite the learning task. However, extra created features combined with the original given ones can contain overlapping information. So, this increase in dimensionality can backfire and reduce the accuracy of the model. To handle this, after feature engineering, we exploit the principal components analysis (PCA) technique [36]. By transforming the features into a new set of uncorrelated variables (principal components), PCA helps to remove redundant or less informative features while retaining the most significant patterns and variances. In short, Mahak utilizes PCA to identify a lower-dimensional representation of the input features that can capture the most important information. Later, this lower-dimensional representation is used as a reduced set of features for subsequent modeling.

4.2 Mahak's AL Block

The Supervised Model: To address the design challenges and achieve a performance map of the target protocol (f), Mahak requires a learning model based on regression. The purpose of this regression model is to learn the mapping from the input network space to the performance of f. Since the performance metric can be any function, Mahak's learning model is defined as regression instead of classification. Additionally, Mahak faces the challenge of working with limited training data and the need to provide a confidence score for predictions, which is necessary for Active Query 3.

Addressing these requirements, Mahak employs the Gaussian Process Regressor (GPR) [33] in its learning block. GPR not only serves as a powerful non-parametric regression model but also provides uncertainty estimates alongside its predictions for each data point. This uncertainty information becomes valuable for the active query block in selecting instances for labeling. Furthermore, GPR demonstrates reliable predictive capabilities even when there is limited labeled data, aligning well with Mahak's challenges.

GPR models the relationship between input features and output variables using a Gaussian process, defined by a mean and a covariance (or kernel) function, which gauges the similarity between input points. Given a dataset D and a new data point with feature u_t and label d_t, GPR estimates the probability as: $P(d_t|x_t, D) \sim \mathcal{N}(\mu, \Sigma)$.

The GPR model predicts using labeled examples, estimating mean and covariance functions from observed data. Key to GPR is the kernel function, with the RBF and rational quadratic kernel being popular choices. Mahak employs the RBF kernel [24], ideal for non-linear data patterns. In the RBF kernel, the similarity between two points u_x and $u_{x'}$ is gauged by the squared Euclidean distance: $K(u_x, u_{x'}) = exp(-\frac{||u_x - u_{x'}||^2}{2\sigma^2})$.

The Active Query Selection: At the core of any AL-based design lies an active query selection algorithm determining which new points from the set U should be chosen and added to the set D. The active query algorithm is the key component that enables AL to achieve its objective of learning a mapping with significantly fewer labeled samples compared to other types of ML algorithms. Mahak utilizes uncertainty sampling as its active query selection algorithm [26].

The fundamental idea behind uncertainty sampling is that the most valuable examples for learning are those that the ML model is least confident about. As a result, the AL cycle focuses its efforts on gathering information about the unlabeled instances that confuse the model, rather than querying instances about which the model is already confident. In the case of Mahak, which employs GPR as a probabilistic regression technique, each data point's uncertainty is determined by measuring the variance of the predicted distribution. Consequently, in each iteration of Mahak, the uncertainty sampling strategy selects an unlabeled instance u from the set U that the GPR regression model is least confident about. After labeling it, the pair $\{u, f(u)\}$ is added to the set D, and the GPR model is retrained using the updated D.

An Automated Oracle: This block is responsible for the annotation of unlabeled data as shown in Fig 1. When the active query block determines the next required setting to be evaluated, the oracle needs to bring up the specified setting, evaluate the protocol, and insert the labeled data to D. In other words, it takes $u \in U$ as the input, evaluates the performance/label, $f(u)$, and inserts $d = (u, f(u))$ to D. In the next iteration, the updated D is utilized for GPR training. Here, Mahak directly employs the actual (and not simulated/simplified) implementation of the protocol to send traffic. This enables Mahak to be protocol-agnostic and not only assess heuristic CC schemes but also complicated ML-based ones. To create the network, currently, Mahak, without loss of generality, uses Mahimahi's emulated links [28]. Algorithm 1 shows how each block functions in Mahak.

Algorithm 1: Mahak Active Learning Algorithm

Initiate $U = (u_1, u_2, ..., u_n)$ `// The unlabeled dataset as combination of all network parameter features`
Apply Feature Extraction and PCA on U
Initiate $D = \emptyset$ `// The labeled dataset defined as empty set at first`
while *Computation Budget not met* **do**
  Select an instance u_j from U which the GPR is most uncertain about `// Uncertainty Sampling`
  Measure the performance (d_j) of the selected instance u_j `// Mahimahi Emulation`
  Add (u_j, d_j) to D `// Update the labeled dataset`
  Retrain the GPR model using D `// Update the model using the labeled dataset`
Output the learned performance mapping by GPR

5 EVALUATION

To highlight the protocol-agnostic aspect of Mahak and the fact that it treats the protocols as a black box, and does not require any assumptions, simplification, or prior explicit mathematical models of them, we choose two CC schemes: BBR2 [12], and Orca [6]. BBR2 is a heuristic scheme, while Orca is a state-of-the-art scheme utilizing complicated deep reinforcement learning techniques. We used modAL [14] to implement Mahak's AL block.

During the evaluations, we use the following boundaries ($[\min, \max]$) for the input features: $[4, 100]$ ms for mRTT, $[10, 20000]$ packets for the bottleneck buffer size, $[10, 100]$ Mbps for the link bandwidth, and $[0.1\times, 8\times]$ for the change ratio of the bandwidth.

5.1 Performance Model and Its Accuracy

We start the evaluations by demonstrating a sample performance map (of a heuristic scheme, BBR2) obtained by Mahak and elaborate on its accuracy. To quantify the model accuracy, we calculate the error between Mahak's output and the

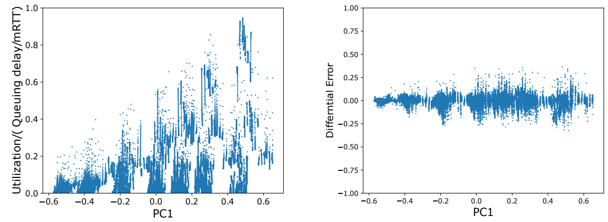

Figure 3: One-dimensional visualization of the performance map of BBR2 obtained by Mahak (left) and its error (right)

Target CC	MAE	95th %tile AE	RMSE	Performance Metric
BBR2	0.02	0.09	0.04	Link Utilization
Orca	0.03	0.11	0.05	Link Utilization
BBR2	0.02	0.09	0.04	Queuing delay / mRTT
Orca	0.008	0.03	0.03	Queuing delay / mRTT
BBR2	0.02	0.07	0.04	Link Utilization / (Queuing delay/mRTT)
Orca	0.02	0.07	0.03	Link Utilization / (Queuing delay/mRTT)

Table 1: Mahak Assessment Error on CC Scheme

true values of the CC scheme's performance across the entire space. In this setting, we only permit Mahak to query about 4% of the total existing data points in the defined evaluation space, U. In other words, the computation budget is set so that $|D| \approx 0.04 \times |U|$. To demonstrate the performance map in the entire defined 4D space, we use visualization techniques. In particular, utilizing PCA, we reduce the 4D space into a single dimension. Note that this does not have any impact on the reported error values. For this experiment, we give Mahak the following non-linear performance metric: $\frac{\text{Utilization}}{(\text{Queuing Delay/mRTT})}$. Results are reported in Fig. 3. Considering the error graph (right plot in Fig. 3), it can be confirmed that Mahak's model achieves a low error rate and therefore it has successfully assessed the scheme's performance. Utilizing the (original 4D) assessment map, designers can easily observe regions where the tested CC scheme performs poorly.

Since Mahak utilizes a regression-based approach, to further elaborate on its performance model accuracy, in the rest of this section, we employ well-established regression metrics such as mean absolute error, and root mean square error. Furthermore, we report the 95th percentile of absolute error, AE, ($|y_i - \hat{y}_i|$) to provide a better picture of the error statistics of Mahak's gained performance map. Table 1 reports these errors for tested CC schemes and two linear performance metrics, the average link utilization and the average normalized queuing delay, and one non-linear metric, $\frac{\text{Utilization}}{(\text{Queuing Delay/mRTT})}$. The low values of errors indicate that independent of different performance metrics, Mahak successfully obtains the performance map of the target CC algorithms across the space while evaluating a small fraction of the scenarios (≈4%).

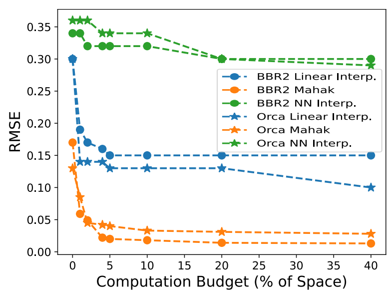

Figure 4: Performance vs. Budget

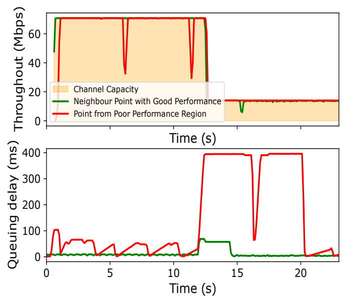

Figure 5: High Delay Issue of BBR2

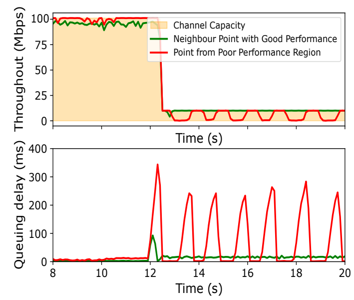

Figure 6: Orca's Oscillatory Behavior

5.2 Impact of Computation Budget

In this section, we discuss how the computation budget influences Mahak's performance. We contrast Mahak with two baselines: nearest-neighbor (NN) interpolation and linear interpolation. NN interpolation assigns the label of the closest labeled data point to an unlabeled point, whereas linear interpolation averages neighboring labels. Fig. 4 shows that increasing Mahak's computation budget decreases the RMSE error. Notably, after covering 5% of the space, Mahak's error becomes minimal, with further budget increases offering slight improvements. This showcases Mahak's effective active learning in minimizing Oracle queries. On the other hand, both interpolation methods have high error even at a 40% budget, indicating that relying on uniformly (randomly) choosing evaluation settings is insufficient.

5.3 Samples of Findings

One of the straightforward benefits of acquiring the performance map of a protocol is that the users can easily observe the settings where the protocol performs poorly. Here, utilizing the output of Mahak, for brevity, we only share two samples of the detected scenarios where CC schemes (a heuristic, BBR2, and a DRL-based one, Orca) behave poorly.

BBR2: Using Mahak's performance map for BBR2, users can pinpoint areas where BBR2 underperforms. For example, the map reveals that when link capacity decreases by a factor of 5 or more, combined with a low mRTT and a large bottleneck link buffer, BBR2 experiences a notable increase in queuing delay. Furthermore, the scheme struggles to quickly mitigate this delay. Fig. 5 presents a sample from this region. For clarity, we also display BBR2's performance at a nearby point. In Fig. 5, red and green curves represent BBR2's behavior in networks with mRTTs of 40ms and 4ms, respectively, both with identical buffer sizes (8000 packets).

Orca: Mahak's protocol-agnostic nature enables users to evaluate complex schemes like Orca. Using this, we identified an area where Orca struggles, particularly displaying oscillations when link capacity drops over 10-fold. Fig.6 depicts a representative point from this area. For clarity, Orca's performance in a nearby but different region is also presented. In Fig.5, the red and green curves show Orca's behavior in networks with 20ms and 8ms mRTTs, both with the same link capacity, variations, and buffer size.

Notice: Answering the question of what causes these performance issues in different regions is beyond the scope of this work. Mahak is designed as an assessment tool to uncover the overall performance map of CC schemes, though having this tool would greatly help protocol designers uncover performance issues and facilitate improvement efforts.

6 DISCUSSION AND THE FINAL NOTE

Applying Mahak to Other Contexts Beyond CC: In this paper we showed the application of Mahak for assessing CC schemes, however, our approach can be utilized for the assessment of other families of network protocols. This is because Mahak is protocol-agnostic and treats protocols as black boxes. By providing the desired input space features, appropriate performance metrics and updating Oracle block, Mahak's application can be extended to other families of network protocols. We leave similar upgrades to apply Mahak beyond CC for future work.

Final Note: In this work, one of our main goals is to demonstrate that the use of machine learning techniques is not limited to the design of network protocols. In fact, we aim to illustrate that the network protocol assessment, a primary requirement for improving our network infrastructures and developing better network protocol designs, can be modernized by utilizing machine learning techniques. To that end, we introduced Mahak, the first tool employing active learning techniques to automate the CC performance assessment task. Mahak is only a first step toward the goal of fully automating the network protocol performance assessment. We hope that it can stimulate further discussions and inspire the development of more advanced automated assessment tools in the future.

REFERENCES

[1] 2023. Mahak Repository. https://github.com/Ppazhooh/mahak. (2023).

[2] Soheil Abbasloo. 2023. Internet Congestion Control Benchmarking. https://doi.org/10.48550/arXiv.2307.10054. *arXiv preprint arXiv:2307.10054* (2023).

[3] Soheil Abbasloo, Tong Li, Yang Xu, and H. Jonathan Chao. 2018. Cellular Controlled Delay TCP (C2TCP). In *IFIP Networking Conference (IFIP Networking), 2018.*

[4] Soheil Abbasloo, Yang Xu, and H. Jonathan Chao. 2019. C2TCP: A Flexible Cellular TCP to Meet Stringent Delay Requirements. *IEEE Journal on Selected Areas in Communications* 37, 4 (2019), 918–932.

[5] Soheil Abbasloo, Yang Xu, H Jonathon Chao, Hang Shi, Ulas C Kozat, and Yinghua Ye. 2019. Toward Optimal Performance with Network Assisted TCP at Mobile Edge. *2nd USENIX Workshop on Hot Topics in Edge Computing (HotEdge 19)* (2019).

[6] Soheil Abbasloo, Chen-Yu Yen, and H. Jonathan Chao. 2020. Classic Meets Modern: A Pragmatic Learning-Based Congestion Control for the Internet. In *Proceedings of the 2020 ACM SIGCOMM Conference*. 16.

[7] Soheil Abbasloo, Chen-Yu Yen, and H Jonathan Chao. 2021. Wanna make your TCP scheme great for cellular networks? Let machines do it for you! *IEEE Journal on Selected Areas in Communications* 39, 1 (2021), 265–279.

[8] Venkat Arun, Mina Tahmasbi Arashloo, Ahmed Saeed, Mohammad Alizadeh, and Hari Balakrishnan. 2021. Toward formally verifying congestion control behavior. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*. 1–16.

[9] Andrea Baiocchi, Angelo P Castellani, and Francesco Vacirca. 2007. YeAH-TCP: yet another highspeed TCP. In *Proc. PFLDnet*, Vol. 7. 37–42.

[10] Lawrence S Brakmo, Sean W O'Malley, and Larry L Peterson. 1994. *TCP Vegas: New techniques for congestion detection and avoidance*. Vol. 24. ACM.

[11] Carlo Caini and Rosario Firrincieli. 2004. TCP Hybla: a TCP enhancement for heterogeneous networks. *International Journal of Satellite Communications and Networking* 22, 5 (2004), 547–566.

[12] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2019. TCP BBR v2 Alpha/Preview Release. https://github.com/google/bbr/blob/v2alpha/README.md. (2019).

[13] Claudio Casetti, Mario Gerla, Saverio Mascolo, Medy Y Sanadidi, and Ren Wang. 2002. TCP Westwood: end-to-end congestion control for wired/wireless networks. *Wireless Networks* 8, 5 (2002), 467–479.

[14] Tivadar Danka and Peter Horvath. [n. d.]. modAL: A modular active learning framework for Python. ([n. d.]). https://github.com/modAL-python/modAL available on arXiv at https://arxiv.org/abs/1805.00979.

[15] Mo Dong, Tong Meng, Doron Zarchy, Engin Arslan, Yossi Gilad, Brighten Godfrey, and Michael Schapira. 2018. PCC Vivace: Online-Learning Congestion Control. In *15th USENIX Symposium on Networked Systems Design and Implementation NSDI 18)*. 343–356.

[16] Sally Floyd. 2003. *HighSpeed TCP for large congestion windows*. Technical Report.

[17] Cheng Peng Fu and Soung C Liew. 2003. TCP Veno: TCP enhancement for transmission over wireless access networks. *IEEE Journal on selected areas in communications* 21, 2 (2003), 216–228.

[18] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. 2017. Deep bayesian active learning with image data. In *International conference on machine learning*. PMLR, 1183–1192.

[19] Sangtae Ha, Injong Rhee, and Lisong Xu. 2008. CUBIC: a new TCP-friendly high-speed TCP variant. *ACM SIGOPS Operating Systems Review* 42, 5 (2008), 64–74.

[20] Mohsen Sheikh Hassani and James R Green. 2018. Active learning for microRNA prediction. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, 5–12.

[21] David A Hayes and Grenville Armitage. 2011. Revisiting TCP congestion control using delay gradients. In *International Conference on Research in Networking*. Springer, 328–341.

[22] Tom Henderson, Sally Floyd, Andrei Gurtov, and Yoshifumi Nishida. 2012. *The NewReno modification to TCP's fast recovery algorithm*. Technical Report.

[23] Van Jacobson. 1988. Congestion avoidance and control. In *ACM SIGCOMM CCR*, Vol. 18. ACM, 314–329.

[24] Motonobu Kanagawa, Philipp Hennig, Dino Sejdinovic, and Bharath K Sriperumbudur. 2018. Gaussian processes and kernel methods: A review on connections and equivalences. *arXiv preprint arXiv:1807.02582* (2018).

[25] Douglas Leith and Robert Shorten. 2004. H-TCP: TCP for high-speed and long-distance networks. In *Proceedings of PFLDnet*, Vol. 2004.

[26] David D Lewis and Jason Catlett. 1994. Heterogeneous uncertainty sampling for supervised learning. In *Machine learning proceedings 1994*. Elsevier, 148–156.

[27] Shao Liu, Tamer Başar, and Ravi Srikant. 2008. TCP-Illinois: A loss- and delay-based congestion control algorithm for high-speed networks. *Performance Evaluation* 65, 6-7 (2008), 417–440.

[28] Ravi Netravali, Anirudh Sivaraman, Keith Winstein, Somak Das, Ameesh Goyal, and Hari Balakrishnan. 2014. Mahimahi: A Lightweight Toolkit for Reproducible Web Measurement. (2014).

[29] Parsa Pazhooheshy, Soheil Abbasloo, and Yashar Ganjali. 2023. A Simple Non-Deterministic Approach Can Adapt to Complex Unpredictable 5G Cellular Networks. *arXiv preprint arXiv:2309.07324* (2023).

[30] Devdeep Ray and Srinivasan Seshan. 2022. CC-fuzz: genetic algorithm-based fuzzing for stress testing congestion control algorithms. In *Proceedings of the 21st ACM Workshop on Hot Topics in Networks*. 31–37.

[31] Daniel Reker and Gisbert Schneider. 2015. Active-learning strategies in computer-assisted drug discovery. *Drug Discovery Today* 20, 4 (2015), 458–465.

[32] Dario Rossi, Claudio Testa, Silvio Valenti, and Luca Muscariello. 2010. LEDBAT: The New BitTorrent Congestion Control Protocol.. In *ICCCN*. 1–6.

[33] Eric Schulz, Maarten Speekenbrink, and Andreas Krause. 2018. A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions. *Journal of Mathematical Psychology* 85 (2018), 1–16.

[34] Burr Settles. 2009. Active learning literature survey. (2009).

[35] Yanyao Shen, Hyokun Yun, Zachary C Lipton, Yakov Kronrod, and Animashree Anandkumar. 2017. Deep active learning for named entity recognition. *arXiv preprint arXiv:1707.05928* (2017).

[36] Jonathon Shlens. 2014. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100* (2014).

[37] Ranysha Ware, Matthew K Mukerjee, Srinivasan Seshan, and Justine Sherry. 2019. Modeling BBR's interactions with loss-based congestion control. In *Proceedings of the internet measurement conference*. 137–143.

[38] Lisong Xu, Khaled Harfoush, and Injong Rhee. 2004. Binary increase congestion control (BIC) for fast long-distance networks. In *Proceedings-IEEE INFOCOM*, Vol. 4. IEEE, 2514–2524.

[39] Chen-Yu Yen, Soheil Abbasloo, and H Jonathan Chao. 2023. Computers Can Learn from the Heuristic Designs and Master Internet Congestion Control. In *Proceedings of the 2023 ACM SIGCOMM 2023 Conference*.