2020 Eighth International Symposium on Computing and Networking Workshops (CANDARW) | 978-1-7281-9919-1/20/$31.00 ©2020 IEEE | DOI: 10.1109/CANDARW51189.2020.00080

RNN-based Approach to TCP throughput prediction

Luyao Bai
University of Tsukuba

Hirotake Abe
University of Tsukuba

Chunghan Lee
Toyota Motor Corporation

***Abstract*—Conventional TCP congestion control algorithms, such as newReno and CUBIC, were designed to avoid bandwidth collapse at times of network congestion. However, the different network scenarios and various network applications associated with congestion would make it difficult to maintain TCP throughput during this time. Thus, we focus on how a recurrent neural network (RNN)-based TCP throughput prediction method can improve throughput under the conditions of congestion to solve this problem. Our proposed method involves training an RNN on the metrics of a TCP connection to predict the TCP throughput. We prepared eight groups of metrics for training, and applied three types of RNNs (i.e., simple-RNN, gated recurrent unit (GRU), and long short-term memory (LSTM)) in our method design. Under our congestion scenarios, a combination of metrics can contribute to improve the throughput. The throughput predicted via our method indicates that it performs better than the TCP CUBIC algorithm. Furthermore, the GRU-based predictive model demonstrated the best performance, achieving 35% higher throughput than the TCP CUBIC algorithm.**

***Index Terms*—Recurrent neural network, TCP congestion control, Machine learning**

I. Introduction

The continuous advancement of network technology and its applications makes the task of designing a congestion control measure difficult, as it must satisfy steadily changing requirements. Furthermore, it is common for multiple network connections to compete for network resources. It is essential to provide a high network throughput; as such, current network research focuses on highlighting the importance of congestion control [1, 2].

Operating systems have several TCP network congestion control schemes, such as newReno [3] and CUBIC [4]. Network metrics, such as the bandwidth, round-trip time (RTT), and packet loss are dependent on the network scale and architecture. It is difficult for the TCP protocols with default TCP parameters to achieve high throughput for every network [5]. Network metrics typically change over time [6, 7]. Thus, time series data can be applied to analyze network congestion [8], however, the following problem remains: how to improve the throughput of the network under the condition that the metrics change over time.

In this paper, we propose an approach to use recurrent neural network (RNN) prediction to increase TCP throughput. The approach involves applying an RNN model and input data selection for the RNN. More specifically, we use an RNN to estimate the congestion window at the sender side. Precise estimation can facilitate adequate control of the congestion window. Before the RNN-based model can yield predictions, the model must be trained. For precise estimation, the metrics used as input data for the RNN model under training should be selected such that they will significantly improve the performance of the RNN [9, 10, 11]. RNN includes some simple networks (e.g., simple-RNN) and some complex networks (e.g., Long short-term memory (LSTM) and gated recurrent unit (GRU)). So it is important to select the most appropriate type of RNN from several options that include a basic Elman network, and a more sophisticated GRU. In this study, we used the Elman network to configure what we have termed the simple-RNN.

We have discussed the importance of performing congestion control to achieve a high throughput, introduced the related problems, and briefly introduced our approach. The main contributions of this study are as follows:

- Simulations were performed to compare and evaluate the performances of the simple-RNN, LSTM, and GRU models with respect to the TCP throughput and packet loss rate.
- Different groups of metrics were applied to determine which group was best suited for the proposed RNN-based approach.

The remainder of this paper is organized as follows. First, we introduce the RNN models used in this study in Section II. We explain our RNN-based TCP congestion control scheme in Section III. Then, we evaluate our approach in Section IV, and discuss related work in Section V. Lastly, we conclude our paper by summarizing the contributions of this study in Section VI.

II. Overview of RNN models

We will introduce all of the RNN models we used in this paper and explain their characteristics. In Fig. 1, X is the input of an RNN unit, and Y is the output. RNN consists of many connected units. $tanh$ and $sigmod$ are two control function. We will not explain the control functions here because of space restrictions.

A. simple-RNN

The simple-RNN is a basic version of the RNN model. As previously mentioned, the Elman network was used to configure this model. The output is directly implemented as a state. Fig. 1 shows a single unit of the simple-RNN. The simple-RNN contains multiple layers of network units. The number of units and layers that make up the simple-RNN are controllable, this is an essential feature of the simple-RNN.

B. LSTM neural network

LSTM has contributed to improve the simple-RNN model. The most significant characteristic of LSTM is that it prevents the occurrence of the vanishing gradient problem that can be

978-1-7281-9919-1/20/$31.00 ©2020 IEEE
DOI 10.1109/CANDARW51189.2020.00080

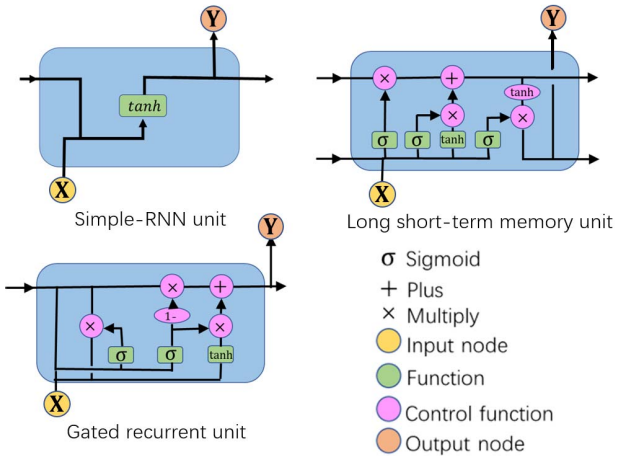


Fig. 1. RNN units

encountered when training conventional RNNs. In this study, we treated the network metrics captured by the congestion control scheme as time-series data; thus, it was difficult for the RNNs to identify long-term time correlations. Fig. 1 also shows the basic structure of an LSTM-based RNN. LSTM can control the memory cell of an RNN because an input gate, output gate, and forget gate are included in the basic structure. These gates allow the rate of gradient descent to be controlled, thereby effectively enabling the long-term processing of associated information.

C. GRU neural network

The GRU is an upgraded version of the LSTM unit. In addition to other modifications, the forget gate and input gate of the LSTM are combined to form an update gate. Its basic structure is shown in Fig. 1, it is simpler than that of the LSTM unit. Specifically, there are approximately one-third fewer parameters, but LSTM's advantage of remembering the gradients during long-term learning is maintained. Because of this advantage, GRUs are less prone to overfitting.

III. PROPOSED METHOD

In this section, we describe the RNN-based prediction method in detail, and discuss the selection of the network metrics for RNN input.

A. System model

Our system can be divided into three components: the congestion control scheme, RNN model, and simulation network. The first component is the congestion control scheme. To control the sender's congestion window, the function collects well-known network metrics in the TCP connection that are then applied as input metrics for the RNN model. Details on the metrics are discussed later in the metrics combinations subsection. The data are transmitted through the network in accordance with TCP. After TCP packet loss occurs, the congestion control component inputs the congestion metrics into the RNN model. The RNN model then makes predictions based on records of the network metrics. The congestion control component uses the prediction to decide how to modify the congestion window.

B. Congestion control scheme

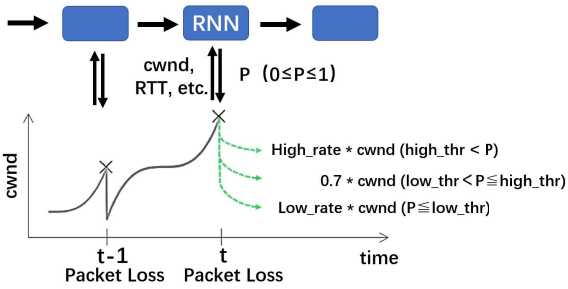


Fig. 2. Approach

Fig. 2 illustrates the RNN-based predictive approach. The X-axis represents the timeline of the changes to the congestion window ($cwnd$). In Fig. 2, congestion occurs at $t-1$ and t, if $cwnd$ at t is larger than that at $t-1$, the training data are marked as 1. Otherwise, it is marked as 0. The trained RNN has an output range of [0, 1]. If the predicted $cwnd$ value is larger than that associated with packet loss, the output value will be close to 1. Otherwise, it will be close to 0. P is used to obtain the output data value, because the value of $cwnd$ is dependent on P.

$$next\_cwnd = \begin{cases} cwnd * low\_rate, P \in [0, low\_pridection) \\ cwnd * 0.7, P \in [low\_thr, high\_pridection) \\ cwnd * high\_rate, P \in [high\_pridection, 1] \end{cases} \quad (1)$$

As described by (1), $next_cwnd$ is dependent on three intervals of P, i.e., $[0, low_prediction)$, $[low_prediction, high_prediction)$, $[high_prediction, 1]$, where $low_prediction$, and $high_prediction$ are manually set values, and a value of P close to 0 means that the predicted value (i.e., $next_cwnd$) is lower than the current $cwnd$ value. For these three intervals, $high_rate = 0.95$, $medium_rate = 0.7$, and $low_rate = 0.5$. $cwnd$ will increase or decrease depending on which interval is applied. When $P < low_prediction$ (i.e., $low_prediction = 0.7$), the sender will quickly reduce $cwnd$ to prevent congestion. When $P > high_prediction$ (i.e., $high_prediction = 0.9$), the predicted value is larger than the current $cwnd$; thus, the algorithm will increase the $cwnd$ to maintain a relatively high transmission rate. Under the condition of $high_prediction > P \geq low_prediction$, the control equivalent of CUBIC is executed. These three values were empirically determined, and will not be discussed here because of paper length restrictions.

C. RNN prediction method

Under the condition of network congestion, the RNN predicts $next_cwnd$ based on the network metrics. Fig. 3 shows

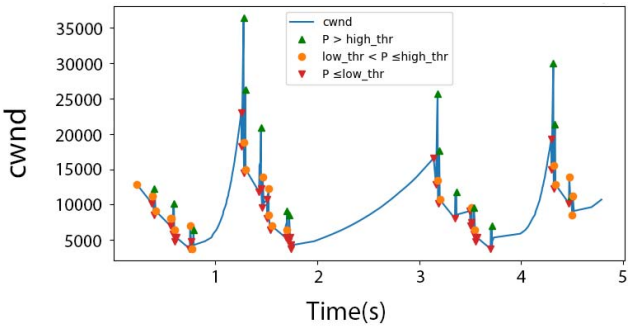


Fig. 3. Network status prediction

that cwnd changed during a TCP data transmission event. As an example, consider a red triangle point in Fig. 3. Before $next_cwnd$ is calculated, we set the prediction is P. P $\in [0; lowprediction)$; thus, the RNN predicts that $next_cwnd$ is low; accordingly, the sender continues to reduce $cwnd$ until the prediction changes. consider a green triangle point in Fig. 3, the value of P in the RNN is higher than that for the high prediction; the sender will continue to increase cwnd until the next P interval is different.

TABLE I
NETWORK METRICS

Name	Explanation
$cwnd$	Congestion window size at packet loss.
$lossint$	Time elapsed since the last packet loss event.
rtt	Round-trip time at packet loss.
$rttd$	fluctuation value of rtt.
$tdup$	Packet loss due to triple duplicate ACKs.
rto	Whether the packet is lost due to retransmission timeout.
$lumax$	$cwnd$ at the time of packet loss that occurred during the CUBIC last Max Probing event.
$luint$	Time elapsed since the last packet loss enent that occurred during max probing.
$ldmax$	Minimum $cwnd$ following continuous decrease of $cwnd$.
$ldint$	Time elapsed since $ldmax$ was last updated.
$tdupint$	Time elapsed since the last Triple Duplicate ACK.
$rtoint$	Time elapsed since the last retransmission timeout.

TABLE II
NETWORK METRICS GROUPS

Number	Explanation
Group 1	$cwnd, lossint, rtt, lumax, luint, tdup, rto$
Group 2	$cwnd, lossint, rtt, lumax, luint, tdup, rto, tdupint$
Group 3	$cwnd, lossint, rtt, lumax, luint, tdup, rto, rtoint$
Group 4	$cwnd, lossint, rtt, lumax, luint,$ $tdup, rto, tdupint, rtoint$
Group 5	$lossint, rtt, lumax, luint, rto, rttd, ldmax$
Group 6	$lossint, rtt, lumax, luint, rto, rttd, ldmax, tdupint$
Group 7	$lossint, rtt, lumax, luint, rto, rttd, ldmax, rtoint$
Group 8	$lossint, rtt, lumax, luint, rto,$ $rttd, ldmax, tdupint, rtoint$

D. Metrics combinations

It is critical to select the appropriate metrics for RNN model training. The network metrics are provided in Table I. Note that max probing is the time required to determine the available bandwidth.

TCP CUBIC uses the RTT and triple duplicate ACKs to regulate cwnd [4]. Triple duplicate ACKs provide clear information on whether the server has requested the right packets, so congestion occurs when a triple duplicate ACK is received. $tdup$ is the number of packets lost because a triple duplicate ACK is required, and rto provides information on whether packet loss occurs because of retransmission timeout. $tdup$ and rto are related to congestion. We can also analyze $lumax$ and $ldmax$ to obtain information about the congestion status. $lossint$ represents the time elapsed since the last packet loss event. We finally prepared eight metric groups in the experiment to evaluate different metrics for input to the RNN models.

IV. PERFORMANCE EVALUATION

In this section, we evaluate the RNN prediction method on the basis of experimental results. We conducted a performance evaluation based on the above-described simulated network. We implemented the packet loss rate and TCP throughput as performance indicators. Consequently, we achieved a 35% throughput improvement relative to TCP CUBIC throughput.

A. Experimental settings

In this section, we introduce our experimental settings. To evaluate the prediction accuracy of the RNN models, we constructed the simulation environment shown in Fig. 4. We employed the RNN-based TCP $sender\_0$. In this study, we individually input the RNN metric value combinations outlined in Table. II for the GRU-based RNN model. There are three types of nodes in this model (i.e., the sender, receiver, and router.) The RNN models were only applied at $Sender\_0$, as $Sender\_1$ was used for cross traffic. The two types of cross traffic applied here were UDP and TCP. The network topology includes six nodes. $Sender\_0$ and $Sender\_1$ are nodes that transmit data, and $Receiver\_0$ and $Receiver\_1$ are the nodes that receive data. $Router_0$ and $Router_1$ are two switch nodes. The bandwidth for all links was 10 Gbps. The cross traffic applied to $Sender\_1$ was set to reach the bandwidth quota for the link to $Receiver\_1$. We also instructed $Sender\_0$ to send 5 GB of data, which was randomly generated by NS-3 (version 3.31) [12], to $Receiver\_0$. Our RNN models were designed based on Keras (version 2.3.1) [13] and Tensorflow (version 2.2.0) [14].

B. Metric Groups

In the experiments, we applied eight different RNN inputs. As can be seen in Table. I, the first four groups mainly consist of $cwnd$, $lossint$, rtt, $lumax$, $luint$, $tdup$, and rto as the main inputs, whereas the last four groups mainly consist of $lossint$, rtt, $lumax$, $luint$, rto, $rttd$, and $ldmax$ as the main inputs. Additional input parameters for the eight test groups were different combinations of $tdupint$ and $rtoint$, and the control group. We found $tdupint$ and $rtoint$ to influence prediction. We selected the eight groups based on the output

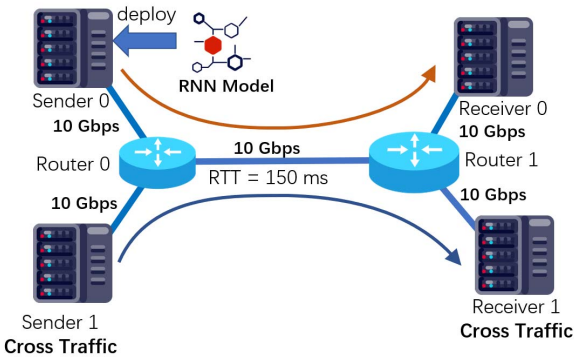
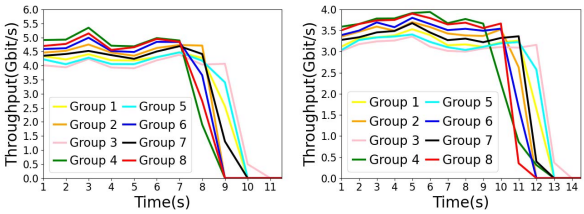


Fig. 4. Network topology for simulation

performance in previous trials. Owing to page restrictions, the combinations of other indicators will not be explained in this paper. Unless explicitly stated otherwise, the RNN model used in this experiment was the GRU-based model, and the input was Group 4.

C. Throughput

The throughput results are shown in Fig. 5 for different network metric groups. The input to the RNN models differed according to metric group. The throughput for the Group 4 TCP cross-traffic and UDP cross-traffic experiments was larger than that for any of the other metric groups. Additionally, the throughput for Group 8 was larger than that for Groups 5, 6, and 7. These results indicate that $tdupint$ and $rtoint$ each have a positive influence.

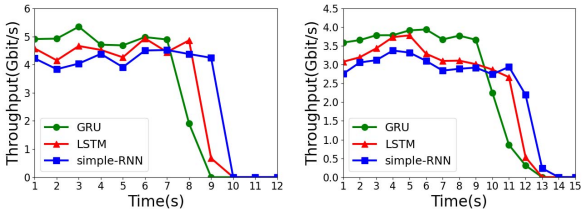


(a) Cross traffic: TCP (b) Cross traffic: UDP

Fig. 5. Throughput for different network feature groups

Fig. 6 illustrates the respective performances of three different RNN models. The throughput for the GRU-based model was better than that for the other two models in this experiment. This finding suggests that, for our TCP prediction method, GRUs are more suitable than the simple-RNN and LSTM.

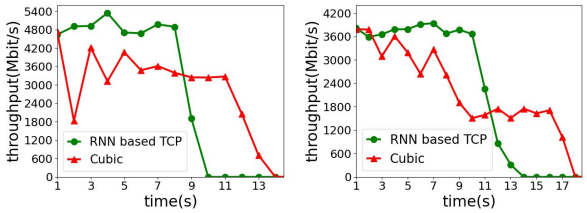
The throughput results for the three RNN models are shown in Fig. 7. In the experiment with TCP as the cross-traffic type, the RNN-based TCP model quickly reached equilibrium, yielding a throughput that was higher than that achieved by CUBIC. Furthermore, the throughput of the CUBIC-based systems wildly fluctuated during the UDP cross-traffic experiment. Thus, RNN-based TCP is not only more stable, but it also yields a throughput that is significantly higher than that of



(a) Cross traffic: TCP (b) Cross traffic: UDP

Fig. 6. Throughput according to RNN model type.

CUBIC. Fig. 7(a) shows that we achieved a 35% throughput improvement relative to the CUBIC model type in the first 8 seconds on average.

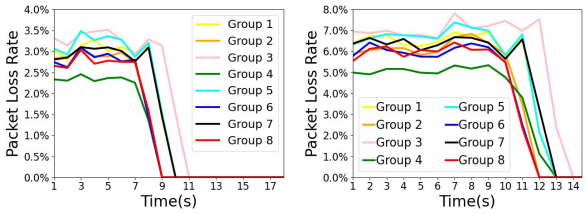


(a) Cross traffic: TCP (b) Cross traffic: UDP

Fig. 7. Throughput comparison between CUBIC and RNN prediction method.

D. Packet Loss

Fig. 8 shows the packet loss rate for different network metrics groups (i.e., different RNN input data). Because we set UDP cross traffic to meet the bandwidth quota, the loss rate was considerably high. Nevertheless, we found Group 4 to have the lowest packet loss rate.



(a) Cross traffic: TCP (b) Cross traffic: UDP

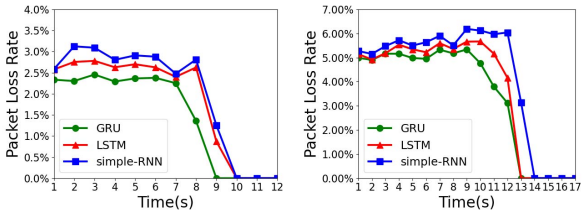
Fig. 8. Packet loss rate for different network metrics groups.

Fig. 9 shows the packet loss rates for the three different models. The GRU-based model had the lowest packet loss rate in both types of cross-traffic experiments.

The results of the TCP cross-traffic experiment revealed that the RNN-based TCP models consistently had lower packet loss rates than their CUBIC-based counterparts (Fig. 10). Comparison of the cross-traffic experimental results also revealed that the RNN-based TCP models enabled faster data transfer.

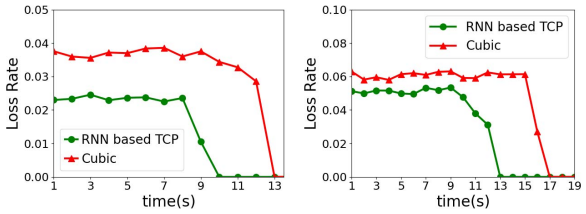
V. Related Work

There are many TCP prediction methods [5, 15] that employ machine learning. For example, there is QTCP [5], which



(a) Cross traffic: TCP (b) Cross traffic: UDP

Fig. 9. Packet loss rate according to RNN model type.



(a) Cross traffic: TCP (b) Cross traffic: UDP

Fig. 10. Packet loss rate comparison between CUBIC and RNN prediction method.

is based on Q-learning [16], and AURORA [15], which is based on reinforcement learning. Various previously developed methods are also based on deep learning, but they do not take into account the influence of input [15, 8]. Alternatively, there have been studies that were conducted to use machine learning to increase the accuracy of TCP prediction; however, these studies did not take into account different machine-learning metrics [17, 18].

VI. CONCLUSIONS

We developed an RNN-based TCP prediction method, and employed different RNN models (i.e., simple-RNN, LSTM, and GRU) to increase TCP throughput. To investigate the impact of different network metric groups on these RNN models, we evaluated them as RNN input data. Our results revealed that a throughput that is approximately 35% higher than that achievable by CUBIC can be realized by employing our proposed prediction method with an appropriate metrics group.

Although it has considerable potential, our RNN-based prediction method can be improved. Toward this goal, we plan to evaluate the fairness among multiple TCP connections in the future.

ACKNOWLEDGEMENT

We would like to express our gratitude to Yosuke Inoue and Takuya Hamasaki for their contributions in the early stage of this research when they were students at the University of Tsukuba. We also thank Toyota Motor Corporation for supporting this work.

REFERENCES

[1] A. Ghaffari, "Congestion control mechanisms in wireless sensor networks: a survey," *Journal of Network and Computer Applications*, vol. 52, pp. 101–115, 2015.

[2] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, no. 1, p. 16, 2018.

[3] L. Vinet and A. Zhedanov, "The NewReno Modification to TCP's Fast Recovery Algorithm," *Journal of Physics A: Mathematical and Theoretical*, vol. 44, no. 8, pp. 1–5, 2011.

[4] S. Ha, I. Rhee, and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," *Operating Systems Review (ACM)*, vol. 42, no. 5, pp. 64–74, 2008.

[5] W. Li, F. Zhou, K. R. Chowdhury, and W. M. Meleis, "QTCP: Adaptive Congestion Control with Reinforcement Learning," *IEEE Transactions on Network Science and Engineering*, pp. 1–13, 2018.

[6] K. Winstein, A. Sivaraman, and H. Balakrishnan, "Stochastic forecasts achieve high throughput and low delay over cellular networks," in *USENIX Symposium on Networked Systems Design and Implementation, NSDI 2013*, 2013, pp. 459–471.

[7] V. Arun and H. Balakrishnan, "COPA: practical delay-based congestion control for the internet," in *Proceedings of the 15th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2018*, 2018, pp. 329–342.

[8] R. Fu, Z. Zhang, and L. Li, "Using LSTM and GRU neural network methods for traffic flow prediction," in *Proceedings - 2016 31st Youth Academic Annual Conference of Chinese Association of Automation, YAC 2016*. Institute of Electrical and Electronics Engineers Inc., 2017, pp. 324–328.

[9] L. Munkhdalai, T. Munkhdalai, K. H. Park, T. Amarbayasgalan, E. Erdenebaatar, H. W. Park, and K. H. Ryu, "An end-to-end adaptive input selection with dynamic weights for forecasting multivariate time series," *IEEE Access*, vol. 7, pp. 99 099–99 114, 2019.

[10] H. Yoon, K. Yang, and C. Shahabi, "Feature subset selection and feature ranking for multivariate time series," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 9, pp. 1186–1198, 2005.

[11] P. Przymus, Y. Hmamouche, A. Casali, and L. Lakhal, "Improving multivariate time series forecasting with random walks with restarts on causality graphs," in *IEEE International Conference on Data Mining Workshops, ICDMW*, vol. 2017-Novem, 2017, pp. 924–931.

[12] "ns-3 — a discrete-event network simulator for internet systems," 2020. [Online]. Available: https://www.nsnam.org/

[13] "Keras: the Python deep learning API," 2020. [Online]. Available: https://keras.io/

[14] "TensorFlow," 2020. [Online]. Available: https://www.tensorflow.org/

[15] N. Jay, N. H. Rotman, P. Brighten Godfrey, M. Schapira, and A. Tamar, "A deep reinforcement learning perspective on internet congestion control," in *36th International Conference on Machine Learning, ICML 2019*. PMLR, 2019, pp. 5390–5399.

[16] C. J. C. H. Watkins and P. Dayan, "Q-Learning," *Machine learning*, vol. 8, pp. 279–292, 1992.

[17] M. Mirza, J. Sommers, P. Barford, and X. Zhu, "A machine learning approach to TCP throughput prediction," in *SIGMETRICS 2007*, vol. 35, no. 1, 2007, pp. 97–108.

[18] B. A. Nunes, K. Veenstra, W. Ballenthin, S. Lukin, and K. Obraczka, "A machine learning approach to end-to-end RTT estimation and its application to TCP," in *Proceedings - International Conference on Computer Communications and Networks, ICCCN*, 2011, pp. 1–6.