Improving TCP Congestion Control with Machine Intelligence

Dr.P.R.C.Murty [1] and Mr.N.Satish Kumar [2]

[1] Professor, Department of Information Technology, MREC (A), Hyderabad-500100
[2] Assistant Professor, Department of Information Technology, MREC (A), Hyderabad-500100

Abstract

In a TCP/IP network, a key to ensure efficient and fair sharing of network resources among its users is the TCP congestion control (CC) scheme. Previously, the design of TCP CC schemes is based on hard-wiring of predefined actions to specific feedback signals from the network. However, as networks become more complex and dynamic, it becomes harder to design the optimal feedback- action mapping. Recently, learning-based TCP CC schemes have attracted much attention due to their strong capabilities to learn the actions from interacting with the network. In this paper, we design two learning-based TCP CC schemes for wired networks with under-buffered bottleneck links, a loss predictor (LP) based TCP CC (LP-TCP), and a reinforcement learning (RL) based TCP CC (RL-TCP). We implement both LP-TCP and RL-TCP in NS2. Compared to the existing NewReno and Q-learning based TCP, LP- TCP and RL-TCP both achieve a better tradeoff between throughput and delay, under various simulated network scenarios

Keywords: LP-TCP, RL-TCP, Congestion Control

I. .INTRODUCTION

Designing TCP congestion control (CC) schemes to ensure efficient and fair use of the network resources has been a well-motivated and intensely studied topic for nearly three decades, resulting in a range of influential algorithms that are either entirely host-to-host, or with in-net support . We focus on host-to-host CC schemes due to their flexibility and independence from the network. Many of the existing host-to-host CC schemes target networks of high-bandwidth and low congestive packet loss rate (e.g., [6, 28]). To support high bandwidth, a rule of thumb is to have the buffer size at each link linearly scale with the link-rate, which causes negative side-effects such as “bufferbloat” (i.e., high latency as a result of excessive buffering of packets) and high hardware cost. Thus reducing buffer size is desirable. It is also shown to have negligible change in throughput when a large number of TCP connections coexist in a single backbone link [1]. However, when the number of coexisting TCP connections is small, an under- buffered (i.e., buffer size smaller than that suggested by the rule of thumb)

bottleneck link can often be under-utilized by existing TCP flows, which reduce their congestion windows (cwnd) frequently upon packet losses. Therefore, the first question we explore in this paper is: Can a TCP CC scheme learn to predict congestive packet losses? Heuris- tics based on the measured throughput or round-trip time (RTT) of a TCP flow perform poorly in loss prediction [. A carefully-built loss predictor model shows higher predic-tion accuracy, but requires sophisticated human design. Recently, capability of machines to learn and represent complex models is re-discovered and exploited to solve various problems in computer networks . Thus, we develop a loss predictor (LP) us- ing supervised learning, and incorporate it into the TCP CC to predict and reduce congestive packet losses. With tuning of a deci-sion threshold th, the loss predictor based TCP (LP-TCP) achieves a desired tradeoff between throughput and delay. Compared to NewReno , a single "always-on" LP-TCP connection shows 29% increase in throughput with similar RTT, in an extremely under- buffered bottleneck link (See Table 5, L = 5). Also, when four LP-TCP connections coexist in an under-buffered bottleneck link, their average throughput increases by 4−5% with slightly increased RTT (See Tables 6 and 7). However, LP-TCP works better when the network model remains more or less fix

II. LITERATURE SURVEY

Distinguishing congestion losses from wire- less transmission losses written by Saad Biaz and Nitin H Vaidya.

The TCP is a popular transport protocol used in the present-day Internet. When packet losses occur the TCP assumes that the packet losses are due to congestion, and responds by reducing its congestion window. When a TCP connection traverses a wireless link, a significant fraction of packet losses may occur due to transmission errors. The TCP responds to such losses also by reducing the congestion window. This results in unnecessary degradation in the TCP performance. We define a class of functions named loss predictors which may be used by a TCP sender to guess the actual cause of a packet loss (congestion or transmission error) and take appropriate actions. These loss predictors use simple statistics on round-trip times and/or throughput, to determine the cause of a packet loss. We investigate their ability to determine the cause of a packet loss. Unfortunately, our simulation measurements suggest

that the three loss predictors do not perform too well

The existing host-to-host CC schemes target networks of high-bandwidth and low congestive packet loss rate. To support high bandwidth, a rule of thumb is to have the buffer size at each link linearly scale with the link-rate, which causes negative side-effects such as “bufferbloat” (i.e., high latency as a result of excessive buffering of packets) and high hardware cost. Thus reducing buffer size is desirable. It is also shown to have negligible change in throughput when a large number of TCP connections coexist in a single backbone link . However, when the number of coexisting TCP connections is small, an under- buffered (i.e., buffer size smaller than that suggested by the rule of thumb) bottleneck link can often be under-utilized by existing TCP flows, which reduce their congestion windows (cwnd) frequently upon packet losses.

III. METHODOLOGY

Now-a-days world is connected with computer networks to pass information to any body and to manage this network activities easily we need to manage all resources such as Routing, congestion control, delay, packet deliver and many more very efficiently. Packets always get delayed if congestion window improperly managed and to avoid this problem Rule based congestion was introduced which check if one path is congested then choose alternate path but this rule based technique will take time to make decision of choosing alternate path. New Reno algorithm is based on Rule based TCP-Congestion management and to further enhance this congestion technique many other algorithms were introduced but their performance is not up to the mark.

In propose paper author is employing machine learning based supervised algorithm called Loss Prediction and Reinforcement Learning algorithms to handle TCP congestion. This algorithms will analyse network state and then learn how to handle congestion and if congestion is learned or predicted then it will choose alternate path. This algorithms will predict or learn congestion very quickly and due to this reason network performance can be increase. Learning quickly can avoid further congestion and delay will get reduced and throughput will get increased.

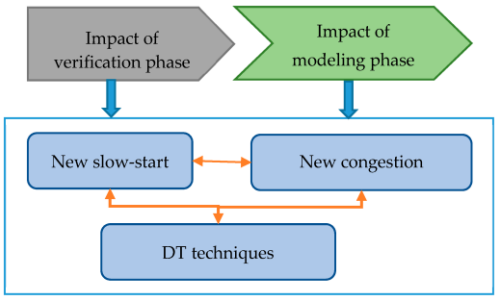


Fig.1. System Architecture

IV. RESULTS AND DISCUSSION

To implement this project we are have designed following modules

1) Upload Network Dataset: using this module we will upload network dataset to application

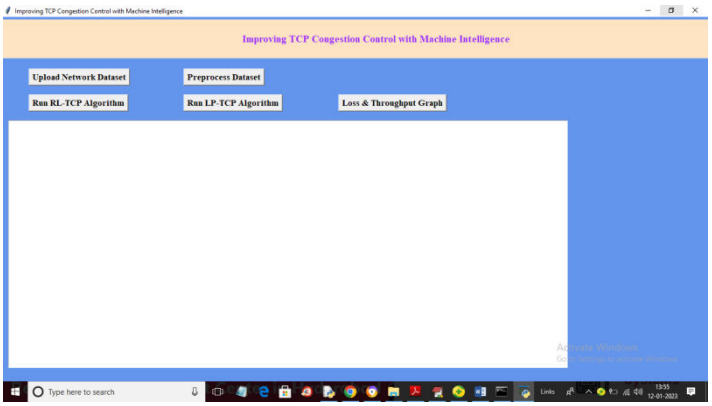
2) Preprocess Dataset: using this module we will read dataset and then remove missing values

3) Run LP-TCP Algorithm: now processed data will be input to LP-TCP supervise algorithmto predict congestion and based on congestion it will predict other path and based on alternate path we will calculate delay and throughput.

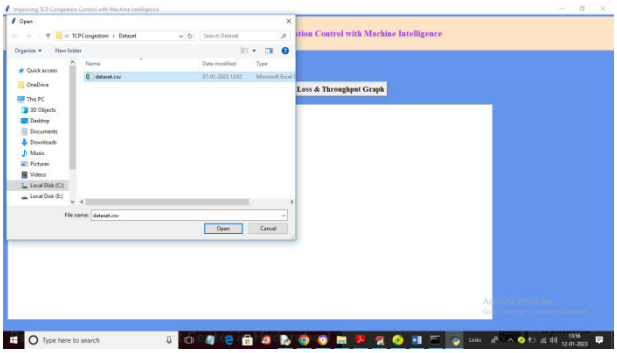
4) Run RL-TCP Algorithm: using this module we will trained RL-TCP algorithm to predict congestion and delay and based on prediction routing get handled and then will calculate delay and throughput. Here we will get delay for existing New-Reno and propose RL-TCP

5) Loss & Throughput Graph: using this module we will plot loss and throughput graph between both algorithms

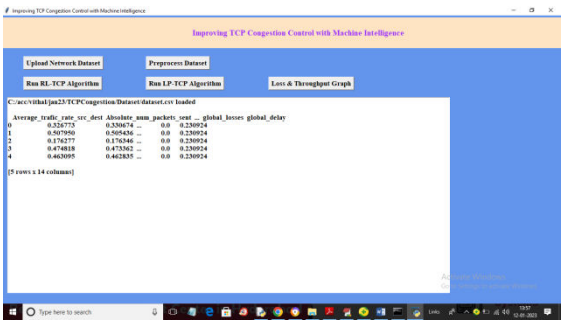
To run project double click on 'run.bat' file to get below screen



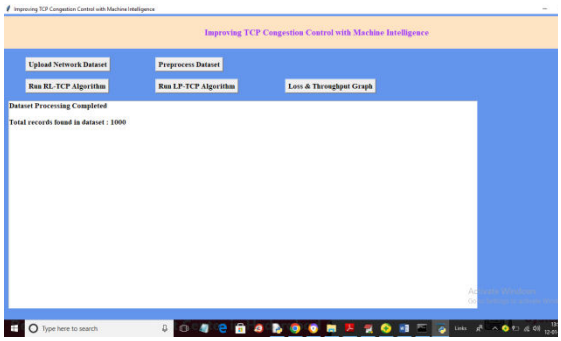
In above screen click on 'Upload Network Dataset' button to upload dataset and get below output



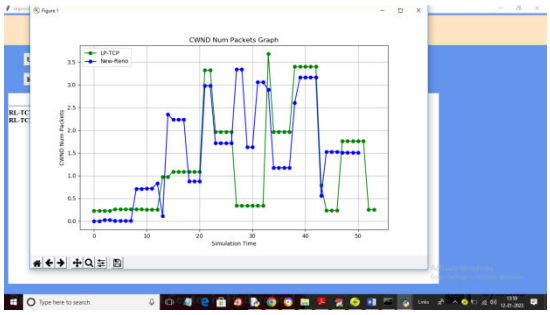
In above screen selecting and uploading dataset and then click on 'Open' button to load dataset and get below output



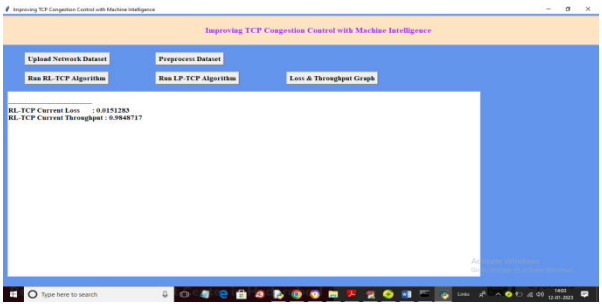
In above screen dataset loaded and now click on 'Preprocess Dataset' button to remove missing values and get below output



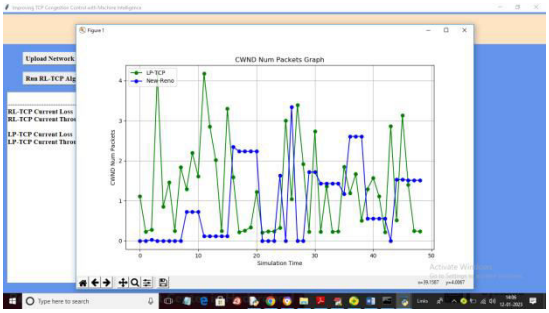
In above screen Preprocessing completed and dataset contains 1000 records and now click on 'Run RL-TCP Algorithm' button to train RL-TCP and get below output



In above graph x-axis represents Simulation Time and y-axis represents CWND window size and for each packet sending we got learning or prediction rate for existing New-Reno (blue colour line) and propose RL-TCP (green colour line) and in above graph we can see RL-TCP got more packet prediction compare to existing New-Reno as RL-TCP prediction time is less so it can process more packets and its throughput will be high. Now close above graph to get below values

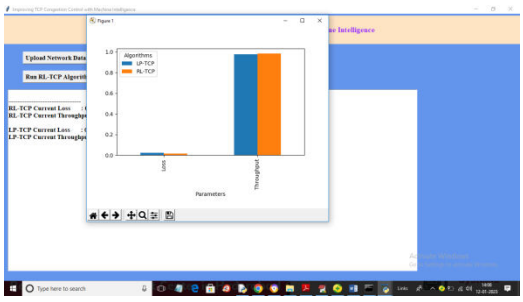


In above screen RL-TCP loss is 0.015 and its throughput is 0.98% as it is processing more packets due to less time in prediction so its throughput will be high and now click on 'Run LP-TCP Algorithm' button to get below output



In above screen we can see LP-TCP (green line) is also better than existing New-Reno to handle congestion and now close above graph to get below screen

In above screen with RL-TCP we got throughput as 0.98 and with LP-TCP we got 0.97 so RL-TCP is better than all other algorithms and now click on 'Loss & Throughput Graph' button to get below output



In above graph x-axis represents algorithm names and y-axis represents LOSS and throughout where orange bar is for RL-TCP and blue bar is for LP-TCP and in both algorithms RL-TCP got high throughput and less LOSS. So we can say with RL-TCP we can improve congestion to get less loss and high throughput

V. CONCLUSIONS

In this paper, we propose two learning-based TCP congestion con- trol schemes for wired networks, one based on supervised learning (LP-TCP), and another based on reinforcement learning (RL-TCP). We evaluate the performance of both schemes in NS2 and compare them to NewReno, Q-TCP, and Qa -TCP. By adjusting a decision threshold, LP-TCP provides a better tradeoff on throughput and delay compared to NewReno. RL-TCP, using our proposed credit assignment, learns effectively in dynamic network environments, and achieves better throughput and/or delay when compared to Q-TCP, Qa -TCP and NewReno at various network configurations. This paper also raises awareness on several issues during the design of learning-based TCP CC schemes. First, the performance of learning-based TCP CC schemes may be parameter-sensitive. For example, the action space of RL-TCP may need a different design depending on the network configurations. It also impacts the com- petitiveness of RL-TCP over NewReno. Second, both learning-based TCP CC schemes can be improved further in achieving fairness among multiple senders in networks with an under-buffered bot- tleneck link.

REFERENCES

[1] Guido Appenzeller. 2005. Sizing router buffers. Ph.D. Dissertation. Stanford University, Palo Alto, CA.

[2] Saad Biaz and Nitin H Vaidya. 1998. Distinguishing congestion losses from wire- less transmission losses: A negative result. In Proc. 7th International Conference on Computer

Communications and Networks. IEEE, Lafayette, LA, 722–731.

[3] Lawrence S. Brakmo and Larry L. Peterson. 1995. TCP Vegas: End to end conges- tion avoidance on a global Internet. IEEE Journal on selected Areas in communica- tions 13, 8 (1995), 1465–1480.

[4] Mo Dong, Qingxi Li, and Doron Zarchy. 2015. PCC: Re-architecting Congestion Control for Consistent High Performance. In Proc. 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI'15). ACM, Oakland, CA, 395–408.

[5] Sally Floyd and Tom Henderson. 1999. The NewReno modification to TCP's fast recovery algorithm. RFC 2582 (1999).

[6] 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.

[7] Tin Kam Ho. 1995. Random decision forests. In Proc. the 3rd international con- ference on Document analysis and recognition. IEEE, Montreal, Que., Canada, 278–282.

[8] Tin Kam Ho. 1998. The random subspace method for constructing decision forests. IEEE transactions on pattern analysis and machine intelligence 20, 8 (1998), 832–844.

[9] Van Jacobson. 1988. Congestion avoidance and control. In Proc. ACM SIGCOMM. ACM, Stanford, CA, 314–329.

[10] Van Jacobson. 1990. Modified TCP congestion avoidance algorithm. note sent to end2end-interest mailing list (1990).