International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 12, Number 12 (2019), pp. 2269-2272

Comparing Different Algorithms based on Reinforcement Learning for Congestion Control

Dr. Mythiliboopathi
Professor,
Vellore Institute of Technology,
Vellore, India.

ORCID: 0000-0001-5957-9860

Rushali Agrawal
Student,
Vellore Institute of Technology,
Vellore, India.

Tanay Arpit Shah
Student,
Vellore Institute of Technology,
Vellore, India.

Abstract

There's a surge in the usage of wired and wireless network, and they've become more and more complex, such that the fundamental assumptions made by the existing TCP variants is not true anymore. Efforts on optimizing the performance of TCP by modifying the core congestion-control technique depending on specific network architectures or apps do not generalize well under a wide variety of network scenarios. This limitation arises from the rule-based design principle, wherever the performance is connected to a pre-decided mapping between the observed state of the network to the corresponding actions. Therefore, these protocols are unable to adapt their behavior in new environments or learn from experience for higher performance. In this paper, we will survey on different techniques of congestion control which is based on reinforcement learning for making decisions and conclude which method is better.

1. INTRODUCTION

With the growth of the internet and mobile networks, the network traffic is an area of major concern today.Even with the increase in capacities of wired as well as wireless links, the gap between what the user demands and what the internet offers is getting wider.

Rapid advancements in wired and wireless technologies have triggered the emergence of new network architectures. With the rapid increase in the number of new applications such as video streaming, cloud storage, on-line gaming, it tends to creates higher performance requirements for the data transmission environment and thereby poses new challenges on the design of congestion control protocols. These online applications are based on TCP performance. User experience and company revenue is directly affected by the performance of TCP which still suffers from an unsatisfactory performance. The main problems that TCP suffers from are:

1. TCP cannot deal with short flows gracefully.
2. The performance of congestion control (CC) algorithms remains far from ideal.

To overcome these issues, a new method is suggested that is, implementation of reinforcement learning methods to dynamically configure IW and CC for improving the network transmission performance in the Internet. Reinforcement learning consists of an agent and an environment, agent is trained to find the optimal solution for a problem and it learns the same with the help of reward. Congestion control is the most important networking function of the transport layer, which ensures reliable delivery of application data. Several congestion control protocols have been designed in the past few years since the advancement in the field of network study and research. With the new protocols being developed, there is a limitation of not being able to perform well in different network sceanrios and hence it is rarely implemented in real world.

2. LITERATURE SURVEY

Author K Xiao et al. [1] suggests a new and a smart congestion control algorithm which is based on Reinforcement Learning implemented by using Deep Neural Networks hence called as Deep Reinforcement Learning. The algorithm is named as TCP – Deep ReInforcement learNing – based Congestion control a.k.a. TCP-Drinc. It learns from the past experience in the form of set of measured features to decide how to adjust the Congestion Window Size. This algorithm's performance is compared to 5 benchmark. algorithms for Congestion Control namely TCP – NewReno, TCP – Cubic, TCP – Hybla, TCP – Vegas, TCP – Illinois. TCP – Drinc achieves the Highest Throughput and 2nd Lowest Round Trip Time (RTT) for increased propagation delay and number of users and bottleneck capacity.

Author Wei Li et al. [2] proposes a novel approach of integrating a reinforcement-based Q-learning framework with TCP design called QTCP. QTCP allows the senders to gradually learn the optimal congestion control policy in an on-line manner. It does not need the hard-coded rules, therefore, it generalize to a variety of different networking scenarios. Furthermore, the author also proposes to develop a generalized Kanerva coding function approximation algorithm, which reduces the computation complexity of value functions. This QTCP protocol can automatically identify the optimal congestion window (cWnd) varying strategy, given the observation of the surrounding networking environment in an on-line manner. Better throughout and delay product than TCP-NewReno (59.5%) and QTCP-Baseline (35.2%). Slightly better RTT performance than NewReno.

The author Xiaohui Nie et .al [3] proposes an improvised algorithm called TCP-RL, which uses reinforcement learning (RL) techniques to dynamically configure IW and CC in order to improve the performance of TCP flow transmission. According to the current research, TCP-RL dynamically configures a suitable IW for short flows through group-based RL, and dynamically configures a suitable CC scheme for long flows through deep RL. In this paper, the author focus on two well-known TCP performance problems:

(1) TCP cannot deal with short flows gracefully

(2) The performance of congestion control (CC) algorithms remains far from ideal.

TCPRL reduces the TCP response time by 23% to 29%. It's performance ranks top 5 for about 85% of the 288 given static network conditions.

3. METHODOLOGY

In [1], Author uses the general setting of congestion control which includes set of remote hosts for serving the mobile users. Multiple hops are required from travelling to the mobile user from the host. The remote hosts apply a congestion window (cWnd) based protocol. They have considered these features as the general settings: competitions among different flows at bottleneck links and potentially multiple bottlenecks along the end-to-end path.

The agent used for reinforcement learning uses the past experience which is stored in the buffer named as experience for the learning purpose. The author used the combination of 3 different methods namely "Feature Selection", "Clipped Rewards", "Modified Training Process" for tackling the problem of multiple agent competition which will lead the agent to wrong states and hence decrease the accuracy.

For simulation purpose, the dropout layer is applied with a 0.2 dropout probability to provide both regularization and ensemble effect. The LSTM layer with 64 units is applied after the DCNN layers. One fully connected layer is used with activation function ELU. The output layer is a linear combination of the previous output with five outputs, one for each action. For simulations, the author has assumed there are N remote hosts and N wired/wireless users i.e. each host has separate agent which do not share information with each other.

The following three testing cases are simulated in this section.

Case I: The number of user are fixed to four and the bottleneck bandwidth to 10 Mbps while the propagation delay is varied between 60 – 240 ms inclusive.

Case II: The number of users is varied in the range of 3-9 inclusive, while the propagation delay is fixed to 80 ms and bottleneck bandwidth to 8 Mbps.

Case III: Author selected the number of user as four and the propagation delay was set to be 100 ms along with that the bottleneck bandwidth is varied between 5 – 20 Mbps inclusive.

In [2], the main goal of congestion control is to allow the sender to share limited bandwidth, without clogging the network. Above function is performed by controlling the congestion window (cWnd) which will limit the number of packets injected by the sender safely without causing clogging. The QTCP model consists of the following elements:

- States: A state is a distinctive profile of the network conditions evaluated through 100 performance metrics. the choice of actions is the key to the QTCP's performance.
- Actions: The action is that the decision to increase, decrease, or leave unchanged the current cWnd
- Rewards: It reflects the desirability of the action picked.
- Training Algorithm: Typically, this can be often the central module of QTCP because it is responsible for developing the congestion control methods.

QTCP works by checking the values of selected state variables and passing these state values to the currently trained policy to generate an action to adjust cWnd. Then QTCP observes the new state and therefore the reward and uses them as an input to the training algorithm that evaluates and improves the cWnd changing policies.

Here, a set of prototype is chosen and used to approximate the value functions, where the state or state-action values are estimated by a linear combination of values of local prototypes. In each time step, only prototypes that are adjacent to the input sample data are updated. adaptive Kanerva Coding: If the set of prototypes is chosen wisely, Kanerva coding works well. If prototypes are not well distributed across the appropriate regions of the state space, several input sample data, in this case, the visited states, will not be adjacent to sufficient prototypes, or maybe worse, they will not have any adjacent prototypes at all to estimate their values. This scenario is caused by poorly selected prototypes and greatly reduces the accuracy valuable selected estimation.

This approach starts with an initial set of randomly selected prototypes and periodically deletes poorly performing/rarely utilized prototypes and generates corresponding new prototypes to bit by bit adjust the allocation of original set of prototypes to cover the region of interest.

In [3], The implementation of TCP-RL includes 3 main modules-

A. connection Manager

B. data Collector

C. Reinforcement Learning

A. Connection Manager: For the initial window configuration for short flow dominated services, the connection Manager queries the IW Table with the user's IP, and therefore the result is the IW for this flow. Then it modifies the IW for this flow immediately. Before the frontend server starts sending data to user, the above mentioned procedures are completed. Once the TCP session is closed, the connection Manager logs the TCP performance data of this session.

For the CC configuration for long flow dominated services, the difference is that the connection Manager queries the Neural Network with the network state and therefore the result is the CC for this flow then, it modifies the CC for this flow. The connection Manager repeats these procedures at a timescale of seconds till the flow transmission is finished.

B. Data Collector: each frontend server outputs {the data| the info| the information} in a log file and conjointly uses hypertext transfer protocol POST to send the data to a centralized data storage platform in data Collector. Reinforcement Learning takes these performance data as the basic input. All the data are collected in real time, and data Collector aggregates and monitors the network. One necessary point to be noted here is that the TCP response time is the key metric for evaluating the performance for short flow.

C. Reinforcement Learning: once data are collected, TCP-RL runs user grouping and RL algorithm. The user grouping algorithm runs at a long scale. The RL algorithm runs at a timescale of minutes to continuously learn the suitable IW or CC scheme for every user group. This module controls all the frontend server's behavior by updating their IW Tables. it is implemented with Golang and Python in the control center.

4. CONCLUSIONS

TCP-Drinc achieves a much lower RTT performance than the loss based protocols e.g. at least 46% lower than TCP-NewReno [1] while QTCP - Generalization outperforms TCP – NewReno by 59.5% [2]. **Hence, we can conclude that QTCP is better than TCP – Drinc.**

TCP-Drinc has much more throughput than TCP-Vega almost double i.e. 100% more than TCP – Vegas, while RTT is only 15% higher for TCP-Drinc [1]. TCP-RL has less throughput than TCP – Vegas (TCP – Vegas has approximately 23.4% more than TCP - RL), although, but it has a very low RTT as compared it, TCP - RL is approximately 11% of TCP – Vegas.

So now, we can find out the Threshold for TCP – Drinc, i.e.

$$Throughput(Drinc) = Throughput(Vegas) + 100\% * Throughput(Vegas)$$

$$= 2 * Throughput(Vegas)$$

$$= 2 * 71.97$$

$$= 143.94\ Mbps$$

So, we can say that, Throughput of TCP Drinc i.e. 143.94 Mbps is more than Throughput of TCP- RL i.e. 58.31 Mbps

$$RTT(Drinc) = RTT(Vegas) + 15\% * RTT(Vegas)$$

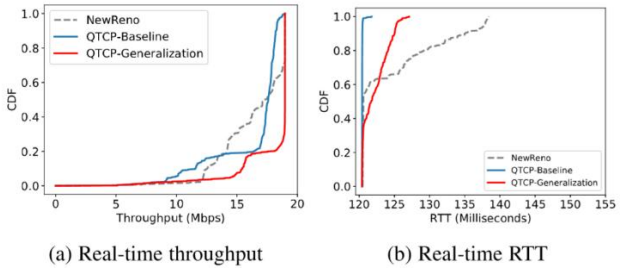
$$= 1.15 * Throughput(Vegas)$$

$$= 1.15 * 110.7$$

$$= 127.305\ ms$$

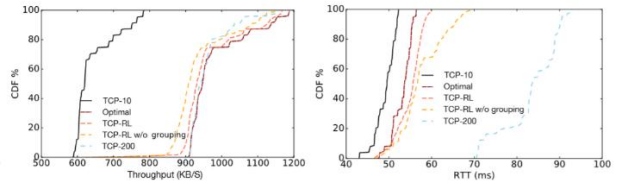
So, we can say that, RTT of TCP Drinc i.e. 127.305ms is more than RTT of TCP- RL i.e. 12.16ms

TCP – RL sacrifices Throughput for getting a better RTT, **hence we can conclude that TCP – Drinc is better than TCP – RL.**



(a) Real-time throughput (b) Real-time RTT

Fig 1. CDF vs Real Time Throughput and Real Time RTT [2]



(c) Throughput (d) RTT

Fig 2. CDF% vs Throughput and RTT [3]

As seen from the graph in Fig.1 QTCP – Generalization achieves 100% CDF at the throughput of 19 Mbps with the RTT as 128ms and from Fig.2 TCP-RL achieves 100% CDF at 1.2 Mbps with RTT as 55ms. We actually cannot compare these values because they are carried out in different conditions. So, for comparing we can find out the ratio of Throughput/RTT for each model. Hence, for QTCP – Generalization, we get the ratio as:

$$\frac{19}{128} * 1000 = 148.43\ Mb/s^2$$

and for TCP-RL, we get the ratio as:

$$\frac{1.2}{5500} * 1000 = 21.81\ Mb/s^2$$

From above we prove, QTCP is better than TCP – RL as the ratio is higher, and we want the throughput to be higher and RTT to be lower.

Hence we can give the following order of Congestion Control Algorithms:

$$QTCP > TCP - Drinc > TCP - RL$$

REFERENCES

[1] Xiao, K., Mao, S., & Tugnait, J. K. (2019). TCP-Drinc: Smart Congestion Control Based on Deep Reinforcement Learning. IEEE Access, 7, 11892-11904.

[2] Li, W., Zhou, F., Chowdhury, K. R., & Meleis, W. M. (2018). QTCP: Adaptive congestion control with reinforcement learning. IEEE Transactions on Network Science and Engineering.

[3] Nie, X., Zhao, Y., Li, Z., Chen, G., Sui, K., Zhang, J., ... & Pei, D. (2019). Dynamic TCP initial windows and congestion control schemes through reinforcement learning. IEEE Journal on Selected Areas in Communications, 37(6), 1231-1247.