Network Packet Management: Addressing Loss and Congestion through Utility Function Optimization

A.Vijay Kumar[1], D.Nagaraju[2], Dr.D.Rajeshwari[3]

[1] Assistant Professor, Department of CSE, Sri Indu Institute of Engineering & Technology, Hyderabad

[2] Assistant Professor, Department of CSE, Sri Indu Institute of Engineering & Technology, Hyderabad

[3] Assistant Professor, Department of CSE, Sri Indu Institute of Engineering & Technology, Hyderabad

Abstract: ***To solve the congestion control algorithm of a dynamic network difficult to determine the appropriate size of the congestion window problem. To improve the traditional congestion control algorithm of the UDP black-box model, the packet loss behavior that is not congested or caused by congestion is distinguished. Optimizing the traditional PCC black-box model based on utility function, and improved PCC-DRL optimization algorithm based on the PCC method were proposed. Compared with the existing mainstream congestion control algorithm, the comparison results show that the application of THE PCC-DRL optimization algorithm improves the dynamic network bandwidth utilization rate 9.67%, reduces the packet loss rate 0.24%, reduces the delay 5.69ms, and improves the queue concurrency 6.73%. These results indicate that THE PCC-DRL algorithm has a good effect on distinguishing the packet loss behaviors caused by non-congestion or congestion in dynamic networks, and has good adaptability and robustness to dynamic conditions and congestion forms.***

Keywords: Utility functions, Deep reinforcement learning, PCC, Congestion control

1. Introduction

With the continuous and rapid growth of the Internet, network performance directly affects the quality of service of data transmission. Once the total demand for resources exceeds the maximum available amount, network congestion will occur. Typical effects include slow transmission speeds, high latency, high loss rates and, in severe cases, network crashes. The goal of congestion control is to dynamically adjust the transmission rate to make full use of network resources under dynamic and complex network conditions. A qualified congestion control algorithm should first sense the network state, then consider the influence of the factors including throughput, congestion window, packet loss rate, delay, fairness, and finally make a fast choice of the optimal transmission rate. Usually, the design of the congestion control algorithm is based on prior knowledge and experience one kind of network environment, but the actual network environment changing, based on the rules and the model of traditional congestion control algorithm can not well adapt to environmental changes, so we need the new breed of dynamic adaptive congestion control algorithm to meet the needs of the guarantee network transmission [1]. The popularity of smart phones has driven the development of mobile cellular networks, and with the rapid development of big data technology, data center network has become a research hotspot. Therefore, many congestion control algorithms for these two networks have emerged. For example, M-TCP [2] and Verus [3] are designed specifically for mobile cellular networks. TCP variants DCTCP [4] and ICTCP [5] are intended to serve data center networks. At the same time, with the rapid development of intelligent algorithms such as deep learning and reinforcement learning, the learning-based network congestion control scheme emerges at the right moment. The characteristics of autonomous learning mode enable the congestion control scheme to be automatically adjusted with the changes of the network environment.

According to different control methods, the congestion control algorithm can be divided into the additive increase multiplicative decrease (AIMD) class represented by NewReno [6], the functional type represented by TCP-Cubic [7] algorithm, the rule aggregation type represented by TCP-compound [8] algorithm and the congestion control algorithm based on reinforcement learning. Through the analysis of the congestion control algorithm mentioned above, we find that the core of the congestion control algorithm lies in the real-time perception of the environment state and the dynamic adjustment of the congestion window (CWND). The traditional congestion control algorithm is usually defined by human experts using prior knowledge, so it cannot be adjusted dynamically. The congestion control algorithm based on reinforcement learning can learn the mapping of historical feedback behavior from the environment without manual adjustment to adapt it to special scenarios. Famous congestion control algorithms based on reinforcement learning proposed at present include Remy [9], PCC [10], BBR [11] and Indigo [12].

The congestion control algorithm Indigo based on imitation learning sets the network scene information as expert knowledge, and the decision network is a single-layer LSTM network, so as to realize the congestion control in the current training scene. However, the performance of the algorithm can only play a superior performance in trained scenarios, so the practical application is limited. In contrast, we find that the congestion control algorithm based on DRL only needs a simple neural network and can achieve better performance than the traditional algorithm according to the historical information of the state combination of multiple time slices before the current time. Therefore, we propose a congestion control algorithm based on DRL called PCC-DRL.

Based on PCC algorithm can't distinguish between the congestion and the loss caused by congestion and quickly adapt to the change of network conditions, the system of sending rate mapping for depth of intensive study, and through the balance of throughput, delay, packet loss rate set a reward function, the use of the depth of the simple neural network for the final strategy approach, Thus, the congestion control algorithm based on deep reinforcement learning is realized. Compared with the traditional TCP CUBIC protocol, the experimental results show that the PCC-DRL algorithm can not only solve the problem that the traditional congestion control algorithm is difficult to adapt to network changes and cannot distinguish packet loss, but also has better congestion control effect and robustness.

2. Overview of PCC Algorithm

Compared with most Congestion Control protocols based on TCP, PCC (Performanc-oriented Congestion Control) protocol [4] is based on UDP, so the algorithm can jump out of various RULES of TCP. After all, in today's complex network environment, TCP can no longer preset enough network events. The congestion control is carried out through the model network, and the PCC algorithm can realize the end-to-end transmission rate control without the need for congestion window size adjustment to complete the congestion control. It is a black box model, and its sender can be decomposed into four parts: data transmission, SACK collection, effect value calculation and transmission rate adjustment. The specific operation process is shown as follows:

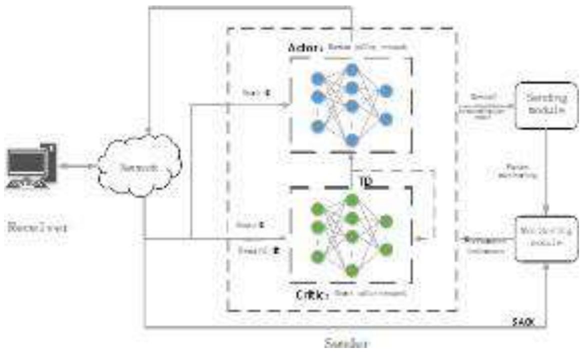
1) Start
In the process of congestion control, PCC algorithm divides time into a series of monitor intervals (MI), and each MI has 1 to 2 RTT. After each MI, the sending rate is doubled, and then the current effect value is calculated according to the current sending rate. When the effect value decreases, PCC will exit the initial stage, but will not directly reduce the sending rate to extremely low like other traditional algorithms, but will look for a higher sending rate of the previous effect value to set.

2) Decision making
As an intelligent congestion control algorithm, its decision strategy is RTCs (Multiple Randomized controlled Trials), which will slightly increase and decrease the current rate R (empirically 0.99r and 1.01R). The four rates of 0.99r, 0.99r, 1.01r and 1.01r were randomly arranged in pairs. After sending, the effect values of these rates were calculated. If the effect values of 1.01r were both large, the sending rate would be set to 1.01r, otherwise 0.99r. However, if there is a staggered situation, the sending rate of R will continue to enter the decision-making stage again. At this time, it is necessary to adjust the coefficient, but keep it between 0.95-0.99 and 1.01-1.05.

3) Rate adjustment
In each MI, the rate will be adjusted according to the sending rate R given in the decision-making stage and the adjustment direction. Only when the effect value of the adjusted rate is smaller than the effect value of the rate before adjustment, the rate will return to the decision-making stage.

3. Improved algorithm: PCC-DRL

PCC uses a black-box approach: THE PCC sender observes the performance metrics generated by sending at a particular rate, converts these metrics into numerical utility values, and adjusts the sending rate in a direction empirically associated with higher utility. However, the specific implementation of PCC shows that the convergence rate is too long and is far from the ideal trade-off between convergence rate and stability.



3.1 Deep reinforcement learning

Applying deep reinforcement learning to congestion control is essentially transforming the congestion control formula into a sequential decision problem under the framework of deep reinforcement learning.

(1) The action is the change of the transmission rate. In congestion control, the agent is the data sender, so its action is actually a change in the transmission rate. We use the concept of monitoring interval (MIs) to define this action concretely. The time is divided into consecutive time intervals. At the beginning of each time interval T, the sender can adjust the sending rate x_t and keep it fixed throughout the time interval. The effects of actions (i.e., rate changes) may have indirect consequences. Sending too fast may overload buffers and lead to future packet loss and delay. In deep reinforcement learning, long-term decisions can be captured by discount factors.

(2) Status is the historical record of network statistics. After the sender selects the ratex_t at the time interval T, it will observe the results sent by the system at this rate and calculate the statistical vector v_t based on the received packets. The statistical vector v_t is composed of three parts: ① Delay gradient (the reciprocal of delay relative to time) ② delay rate (the ratio of the average delay of the current time interval to the minimum average delay of any time interval in the connection history) ③ Send rate (the ratio of packets sent to packets confirmed by the receiver). Networks vary greatly in available bandwidth, latency and loss rate, so when selecting the elements of statistical vector, statistical information that may change greatly between connections,

such as the change of link attributes (absolute value of delay), is avoided, so as to achieve the universality of the model. The agent's choice of the next rate change depends on the fixed-length history of the statistical vector collected from the packets sent by the receiver. This allows agents to detect trends and changes in network state and respond more appropriately, taking into account the limited length of history, not just the latest statistics. Therefore, the states$s_t$at t is defined as: $s_t = (v_{t-(k+d)}, \cdots, v_{t-d})$. For constant k > 0, d represents the delay between the selected sending rate and the collection result.

(3) Set rewards. The rewards generated by sending at a particular rate at a particular time depend on the performance requirements of the particular application. Some applications such as online games require very low latency; Some applications such as large file transfers require high bandwidth; Some services want to use low but constant bandwidth (that isno jitter); Others require higher bandwidth and tolerate bandwidth variations. Therefore, rewards can be better structured according to the performance required by different services. We will implement intelligent congestion control algorithm based on deep reinforcement learning (DRL). DRL is used to generate policies for mapping observed network statistics such as latency, throughput, and so on to rate selection.

3.2 System input and output

Deep reinforcement learning agents essentially solve sequential decision problems by interacting with the environment. In discrete time $t \in 0,1,\cdots$, the agent will observe the perceptible environmental state s_t and select the action a_t according to the current state. Next, the agent will judge whether the action at time t is good or bad according to the feedback reward r_t of the system, so as to make more appropriate action at time $t+1$. The goal of an agent as a whole is to select a strategy π that maximizes the payoff of the action, and deep learning is the best way to approximate the optimal strategy. In the network congestion control system, the action of the agent can be mapped to the transmission rate of the system. According to the statistical vector discussed above, the action of the agent can be mapped to the change of the transmission rate x_{t-1} through function transformation, and the specific change can be judged by the following formula:

$$x_t = \begin{cases} x_{t-1} * (1+\alpha a_t) & ; a_t \geq 0 \\ x_{t-1} / (1-\alpha a_t) & ; a_t < 0 \end{cases} \quad (1)$$

Where α is the scale factor used to suppress the oscillation (α =0.025), a_t is the action performed, x_t is the transmission rate, and the above equation is the transformation formula that maps the action to the transmission rate. When $a_t \geq 0$, it indicates that the current rate has not reached the maximum throughput, and the transmission rate can be further increased to improve the transmission efficiency of network data. Therefore, multiplication mapping is performed to improve the network utilization rate. When $a_t<0$, it indicates that the current rate is too high, and congestion occurs in the system at this rate. Therefore, division mapping is performed to reduce the transmission rate to ensure smooth network.

3.3PCC-DRL

In order to make it be able to recover experience like DQN and make use of previous data, PCC-DRL algorithm adopts importance sampling, so that even the data generated by another strategy can be applied to the training of the current strategy. The factor of importance sampling is actually the ratio of the current strategy to the previous strategy:

$$r_t(\theta) = \frac{\pi_\theta(a_t,s_t)}{\pi_{\theta old}(a_t,s_t)} \quad (2)$$

Which $\pi_\theta(a_t, s_t)$ refers to the current policy, and $\pi_{\theta old}(a_t, s_t)$refers to the last round of strategy.

Then we can obtain its strategic gradient descent algorithm:

$$J_{\theta old}(\theta) = E_{(s_t,a_t)\sim\pi_{\theta old}}\left[\frac{\pi_\theta(a_t,s_t)}{\pi_{\theta old}(a_t,s_t)} A_{\pi_{\theta old}}(s_t,a_t)\right] \quad (3)$$

However, if the difference between the old and new strategies is too large, the system will be unstable due to too much update. The policy difference here refers to that the probability distribution of the actions obtained by the network with the same input state cannot be too different. The similarity degree can be calculated by using THE KL divergence. If the KL divergence value is too large, the punishment will be increased, and if the KL divergence value is too small, the punishment will be reduced. Therefore, the PC-DRL algorithm is shown in the following formula:

$$J_{\theta old}^{PCC-DRL}(\theta) = E_{(s_t,a_t)\sim\pi_{\theta old}}\left[\frac{\pi_\theta(a_t,s_t)}{\pi_{\theta old}(a_t,s_t)} A_{\pi_{\theta old}}(s_t,a_t)\right] - \beta kL(\theta, \theta old) \quad (4)$$

The flow of the PC-DRL algorithm is shown as follows:

```
Input: α,t,rtt
Output: pacing_gain
1  for iteration = 1,2,… do
2  for actor = 1,2,…,N do
3  Run $T$ times in the environment according to the strategy$\pi_{\theta old}$
4  Calculate the average estimate $\hat{A}_1,\cdots,\hat{A}_T$
5  end for
6  The punishment size was adjusted according to the KL divergence value and the processing strategy θ, and the minimum batch M ≤ NT
7      $\theta_{0ld} \longleftarrow 0$
8  end for
```

4. Analysis of Experimental Results

The performance of PCC-DRL algorithm was tested on Pantheon and NS3, and the network topology is shown in the figure. S0 to S1 are the sender, D0 to D1 are the receiver, and they are connected to R1 and R2 respectively. The link bandwidth, delay, and queue management mechanism are 10 Mbps, 1ms, and DropTail. The link between R1 and R2 is a bottleneck link, with bandwidth and delay of 1 Mbps and 20ms. The File Transfer Protocol (FTP) is used as an example to simulate network congestion. The sender uses different TCP data streams. Vegas is used for S0 and Reno is used for S1. The simulation starts at 0s, and the simulation period is 10 s, in which 1% of the sent packets are discarded randomly.

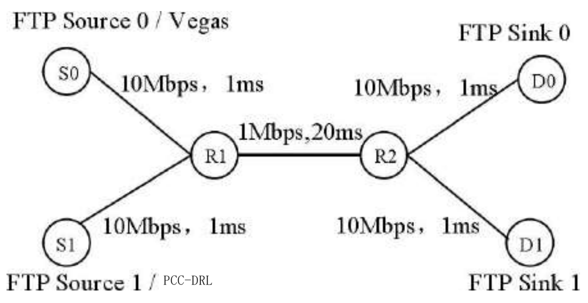


Figure 4.1: Network topology

According to the figure 4.2 we will find that the TCP Vegas agreement under the condition of the link, the throughput is extremely low, thus the random packet loss will also result in system selection will send rate by half, it shows that the traditional congestion control algorithm can't distinguish congestion caused by packet loss or a congestion caused by packet loss, so it cannot make full use of the bandwidth of the link. In contrast, our intelligent system can effectively distinguish the two types of loss by the relationship between packet loss and sending rate. Random lost packets are not affected by the sender's operation, but packet loss caused by congestion will increase with the increase of sending rate. When random packet loss occurs, it increases the transmission rate, but when packet loss is due to link congestion, it does not increase the transmission rate to avoid exceeding link capacity. Therefore, our system can maintain a higher link utilization rate, better use of resources to achieve better data transmission effect.

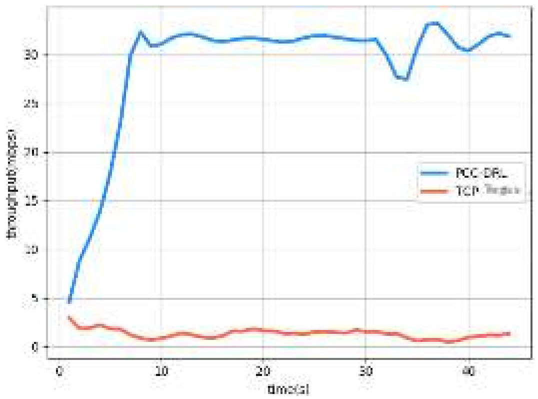


Figure 4.2: Differentiates non-congestion loss from congestion loss

We set the capacity of a single stream on the network link to alternate between 20Mbps and 40Mbps every 5 seconds with no random loss. Ideally, the congestion control protocol changes the transmission rate as it alternates to match the bandwidth of the link to maximize capacity utilization without causing congestion. In the test, we found that TCP Vegas protocol basically cannot adjust the transmission rate with the rate change after 30 seconds. But our intelligence system steep rises suddenly in packet loss can be found that the bandwidth suddenly reduced to reduce the transmission speed quickly, and at the rate of more than 20 MBPS no packet loss when the system can find the available bandwidth higher and quickly adjust to a higher rate of sending 40 MBPS, balance the congestion control and capacity utilization, therefore, Thus, it ADAPTS well to the changing network conditions.

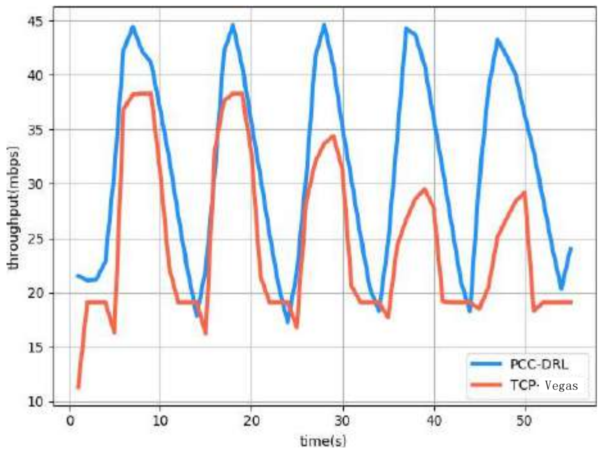


Figure 4.3: Adaptation to changing network conditions

We tested the robustness of the model in the context of link configuration in NS3 with bandwidth between 1Mbps and 1000Mbps, delay of 1ms~256ms, queue of 1000 packets and random loss of 0~ 5%. As shown in Figure 4.4 and 4.5, in general, the trained intelligent network model can well adapt to the network environment with bandwidth changes in terms of delay and link utilization, and achieve better congestion control.

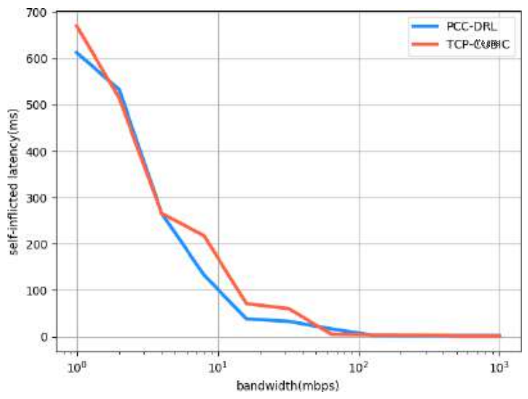


Figure 4.4: Bandwidth sensitivity - delay change

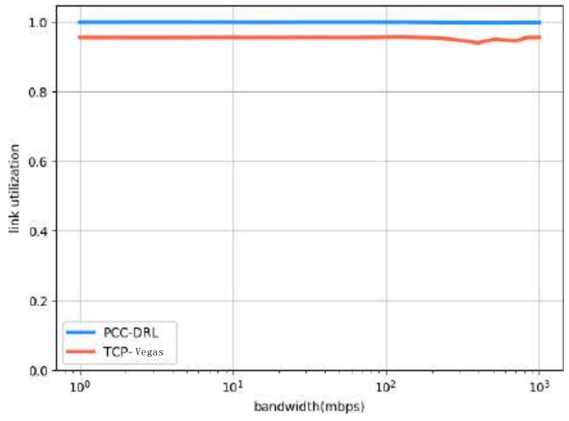


Figure 4.5: Bandwidth sensitivity - link utilization change

And the link utilization, although when the delay is more than 30 ms, intelligent system also can appear a certain degree of link utilization rate of decline, but compared with the bluffs overlooking TCP Vegas protocol type, still can ensure higher throughput, this shows that our algorithm is to balance the throughput, latency and packet loss rate of the system of feedback.

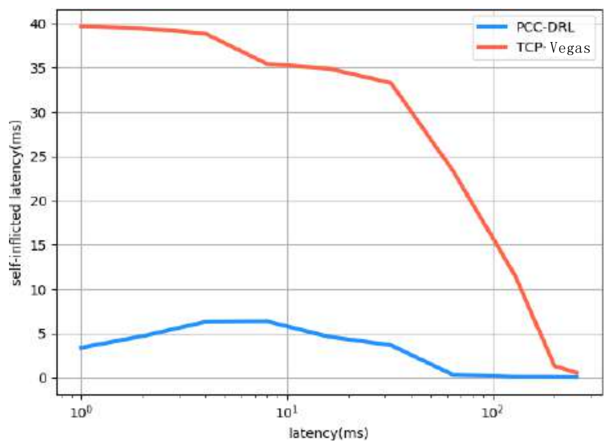


Figure 4.6: Delay sensitivity - Delay change

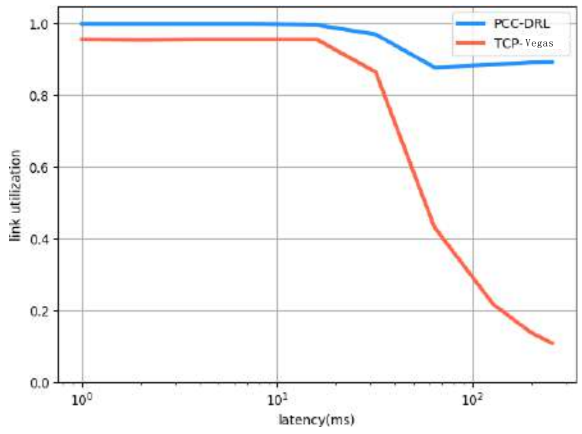


Figure 4.7: Delay sensitivity - link utilization change

In terms of queue sensitivity, the trained intelligent network model can basically guarantee a delay of less than 10ms in terms of delay, and maintain a ratio of about 90% in terms of link utilization. Compared with TCP Vegas, the congestion control effect is greatly different under different queue sizes. It achieves basically stable congestion control effect in the whole queue size range, indicating that it can well adapt to the network environment with queue size variation.

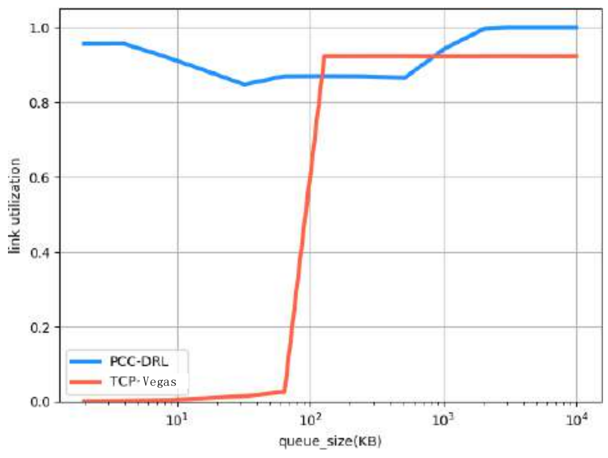


Figure 4.8: Queue sensitivity - link utilization change

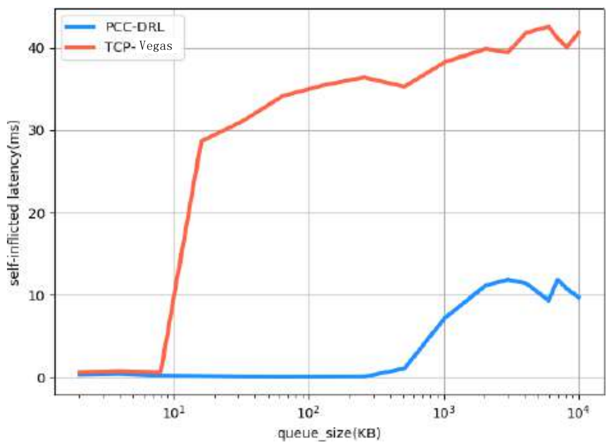


Figure 4.9: Queue sensitivity - delay variation

In the packet loss sensitivity test, our model can achieve the same low latency as TCP Vegas protocol, but the link utilization ratio is much higher than TCP Vegas protocol. TCP is a reliable protocol. Under the traditional congestion control protocol, the 1% packet loss rate directly triggers congestion control and reduces the transmission rate rapidly. However, as mentioned above, intelligent congestion control protocol can distinguish packet loss caused by non-congestion from that caused by congestion. Therefore, random packet loss set by network link basically does not affect the transmission rate of the system. Therefore, even if the packet loss rate reaches 5%, the system can guarantee the link utilization rate of 80%.

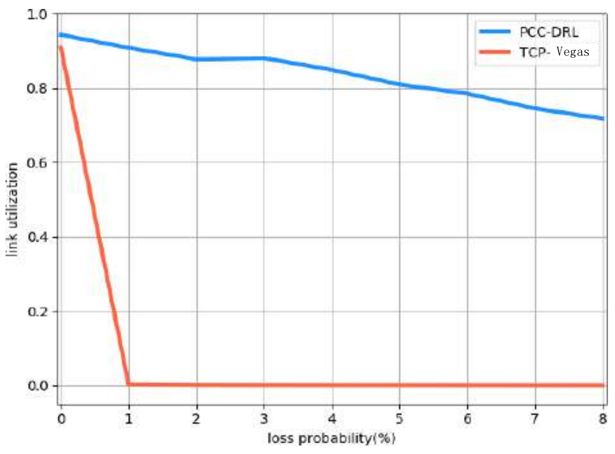


Figure 4.10: Packet loss sensitivity - link utilization change

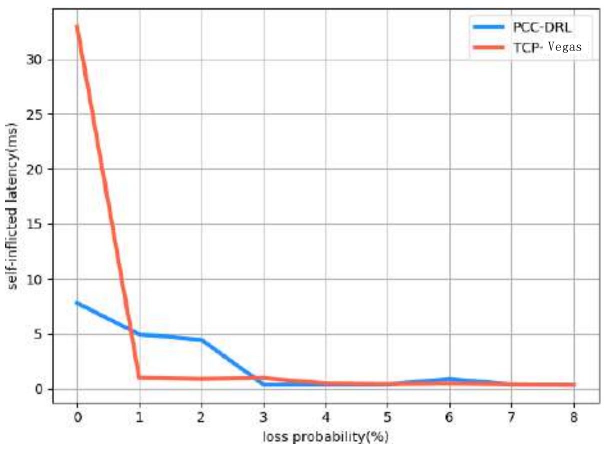


Figure 4.11: Packet loss sensitivity - delay change

5. Conclusion

Based on the research of PCC congestion control protocol, an improved algorithm, PCC-DRL, is proposed to solve the problem that the original PCC algorithm cannot distinguish non-congestion from congestion loss. The PCC-DRL algorithm maps the transmission rate of the system to the action of deep reinforcement learning, sets the reward function by balancing throughput, delay and packet loss rate, and uses a simple deep neural network to carry out the final strategy approximation, so as to realize the congestion control algorithm under deep reinforcement learning.

The simulation results of NS3 show that compared with TCP Vegas, the intelligent system can solve the two drawbacks of the traditional congestion control algorithm: distinguish the loss caused by non-congestion and adapt to the changing network conditions; At the same time, it has excellent robustness under various link conditions such as bandwidth, delay, queue size and packet loss.

In future work, will be further complicated network simulation, continue to optimize the congestion control algorithm based on depth of intensive study, on the other hand, continue to follow up new congestion control algorithm and its iterative update optimization algorithm is applied to other network congestion control algorithm and application scenario, the promotion network congestion control algorithm performance and practicability.

References

[1] S. H. Low, F. Paganini, and J. C. Doyle, "Internet congestion control," IEEE Control Syst., vol. 22, no. 1, pp. 28–43, Feb. 2002.

[2] K. Brown and S. Singh, "M-TCP: TCP for mobile cellular networks," ACM SIGCOMM Comput. Commun. Rev., vol. 27, no. 5, pp. 19–43,1997.

[3] Y. Zaki, T. Pötsch, J. Chen, L. Subramanian, and C. Görg, "Adaptive congestion control for unpredictable cellular networks," in Proc. ACM Conf. Special Int. Group Data Commun., vol. 45, no. 4, 2015,pp. 509–522.

[4] M. Alizadeh et al., "Data center TCP (DCTCP)," ACMSIGCOMM Comput. Commun. Rev., vol. 41, no. 4, pp. 63–74, 2011.

[5] H. Wu, Z. Feng, C. Guo, and Y. Zhang, "ICTCP: Incast congestion controlfor TCP in data-center networks," IEEE/ACMTrans. Netw. (ToN), vol. 21,no. 2, pp. 345–358, Apr. 2013.

[6] Sikdar B, Kalyanaraman S, Vastola K S. Analytic models for the latency and steady-state throughput of TCP Tahoe, Reno, and SACK [J]. IEEE/ACM Transactions On Networking, 2003, 11(6): 959-971.

[7] Ha S, Rhee I, Xu L. CUBIC: a new TCP-friendlyhigh-speed TCP variant [J]. ACM SIGOPS operating systems review, 2008, 42(5): 64-74.

[8] Tan K, Song J, Zhang Q, et al. A compound TCP approach for high-speed and long distancenetworks [C]//Proceedings-IEEE INFOCOM. 2006.

[9] Keith Winstein and Hari Balakrishnan. 2013. TCP ex machina: computer-generated congestion control. SIGCOMM Comput. Commun. Rev. 43, 4 (October 2013), 123–134.

[10] Dong M, Li Q, Zarchy D, et al. {PCC}: Re-architecting congestion control for consistent high performance [C]//12th {USENIX} Symposium onNetworked Systems Design and Implementation ({NSDI} 15). 2015: 395-408.

[11] Cardwell N, Cheng Y, Gunn C S, et al. BBR: congestion-based congestion control [J]. Communications of the ACM, 2017, 60(2): 58-66.

[12] F. Y. Yan, J. Ma, et al., "Pantheon: the training ground for internet congestion-control research," in ATC, 2018.