Applying the Linear Neural Network to TCP Congestion Control

Lei Niu

School of Computer and Information Engineering, Fuyang Teachers College, Fuyang, Anhui Province, China

hefeiniulei@163.com

Keywords: Linear Neural Network, Congestion Control, Decision Boundary, TCP Reno.

Abstract. The previous TCP protocol cannot predict congestion. Only when the sender receives more than three acknowledgements or the retransmission timer is out can it realize that congestion has occurred. We train the linear neural network by using round-trip time and current TCP throughput as its inputs. As a result, we get the decision boundary, which could predict whether the current network is in congestion or not. Simulation results show that, when applied to TCP congestion control, it can effectively predict the occurrence of congestion, so congestion window could make adjustments as soon as possible to reduce the probability of congestion collapse.

Introduction

In recent years due to the growing popularity of Internet, data traffic in computer networks is increasing rapidly. Network congestion has become a serious problem in the networks. Traditional TCP Reno congestion control algorithm only makes response by reducing the congestion window (CWND) when network congestion has occurred, and the data rate also decreases with it at the same time [1]. For better congestion control many scholars have done a lot of meaningful work [2-5]. In this paper a congestion prediction method is proposed in the senders. It can predict congestion according to the current RTT and TCP throughput. So the sender can make adjustments to CWND before congestion collapse to reduce its occurrence.

TCP Congestion Control

TCP congestion control generally contains four parts which are slow-start, congestion avoidance, fast retransmission and fast recovery [6]. After TCP connection is established, slow-start algorithm begins to execute firstly. When the value of CWND has exceeded slow-start threshold, then it enters congestion avoidance stage; if the sender receives three continuous ACKs, it realizes that congestion has occurred. Then the sender begins to retransmit the lost package without waiting for the overflow of retransmission timeout. This is so-called fast retransmission. The next congestion avoidance begins to execute fast recovery rather than slow-start.

TCP finds congestion by receiving continuous ACKs or retransmission timeout overflows. TCP can confirm congestion in either way, but cannot predict it before its occurrence. What's more, the congestion window begins to decrease once it occurs, and the throughput starts to decrease. When congestion becomes serious the congestion collapse is likely to occur and the network throughput drops dramatically at the same time, which affects network performance seriously. So it's very important to predict congestion as early as possible.

Samples Selection

For any TCP connection, if the network congestion occurs, the RTT of TCP connection increases rapidly, meanwhile the TCP throughput decreases. So we select the RTT and the throughput of current TCP as the two inputs of the neural network. In order to obtain training samples, we use the network simulation tool OPNET10.0. The network topology is shown in Fig. 1. R1 and R2 in the topology are two routers. We select ppp_E1 as the link between the router R1 and the router R2 in the simulation, and the highest data rate of the link between the two routers is 2Mbps. But we select the 10BaseT as

the other four links, the its highest data rate is 10Mbps. So congestion would only occur between the router R1 and the router R2.

A sends the Ftp data to B by using the TCP protocol, video conference takes place between C and D by using the UDP protocol. At the beginning of the simulation only A sends data to B, and after 20 seconds C starts to send UDP data to D. UDP data rate also continues to increase every 20 seconds until the congestion occurs.

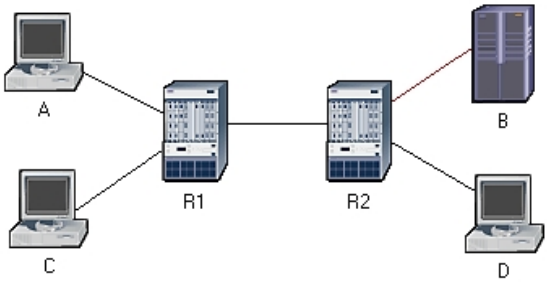

Fig.1 The network topology

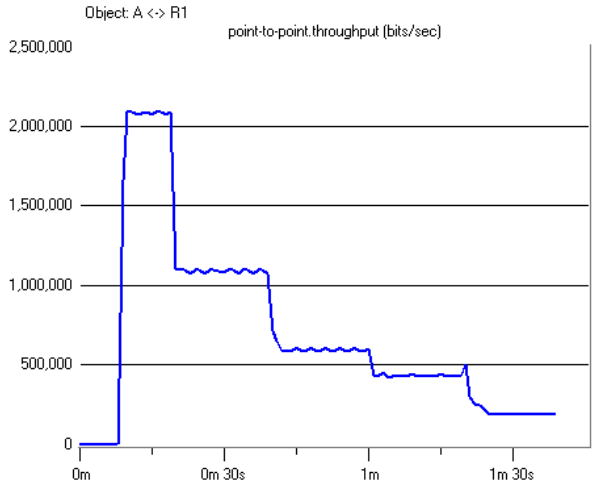

Fig.2 The throughput of TCP

The TCP data rate between A and B as shown in Fig. 2, and the RTT of the TCP connections as shown in Fig. 3. We can find the TCP rate decreases along with the increasing of the UDP rate, meanwhile the RTT value increases with the decreasing of the TCP rate. We then export data and get a total of 90 samples. These samples are from the non-congestion and congestion state. Eventually we select 12 samples from them. The top 6 samples are from the non-congestion state, and the other 6 samples are from the congestion state. As shown in Fig. 4, these 12 samples are normalized. The horizontal axis represents the RTT, and the vertical axis represents the TCP throughput. The 6 coordinate points got in the non-congestion state are represented by the symbol □, and the other 6 coordinate points got in congestion state are represented by the symbol □.

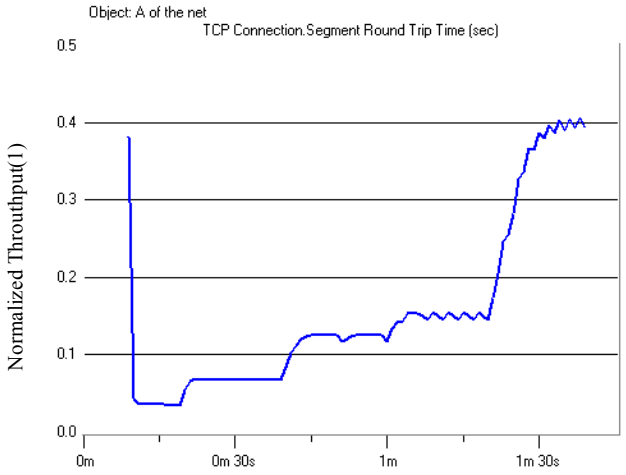

Fig.3 The RTT of TCP

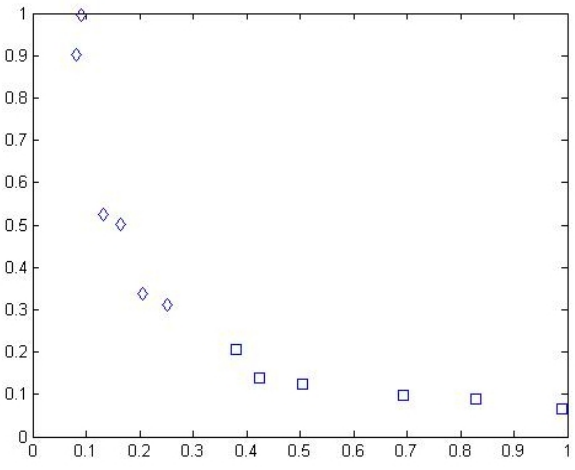

Fig.4 12 selected samples

Network Training

The linear neural network can not only produce binary output, but also generate analog output. The output of linear transfer function can be of any value. Linear neural network diagram as shown in Fig.

5, and x is the input, w is weight, y is the linear output, and q is a binary output. Next we use RTT and the current throughput of TCP connections as the inputs of the linear neural network, and train the network to get the decision boundary used to predict the current congestion state.

We use LMS learning algorithm for the linear neural network. We also name it as Delta rule. The procedure is as follows:

- Step 1: Defining variables and parameters;
- Step 2: Initialization. Giving a smaller initial random value to the weight;
- Step 3: Inputing the samples and calculating the difference of the desired output and the actual output;

$$W(n+1)=w(n)+\eta X^T(n)d(n). \quad (1)$$

- Step 4: Adjusting the weights. Continuing to calculate Eq. 1 according to the errors calculated in the above step. In the Eq. 1, w (n) is weight, η is the learning rate, $X^T(n)$ is the sample inputted, and d(n) is the difference between the expected value and the actual value:
- Step 5: Ending if it's converged.

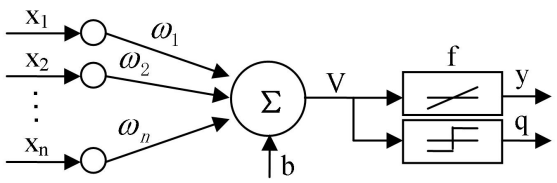

Fig.5 Linear neural network diagram

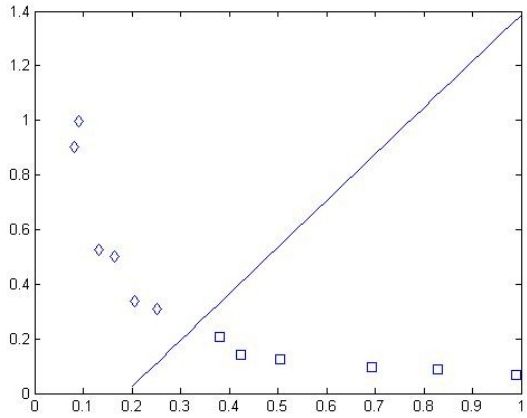

Fig.6 Decision boundary for predicting congestion

$$y=1/2*w(2)-w(1)/w(2)*x-w(3)/w(2). \quad (2)$$

According to the steps above, the maximum learning rate η is calculated firstly. Then 12 samples are inputted. The desired output values are set as six continuous 0 and six continuous 1. Number 0 indicates non-congestion and 1 indicates congestion. Then we start to train the linear neural network. At last the training result w (n) is substituted into the Eq. 2.

1/2 in the Eq. 2 is a threshold of congestion. Eq. 2 is the decision boundary we get for predicting congestion, and the result as shown in Fig. 6. If the coordinate of the RTT and the throughput of the current TCP connection is above the decision boundary, it indicates that the current network state is good and congestion does not occur. On the contrary, if the coordinate is below the decision boundary, it shows that congestion is happening or will happen. If so, the CWND value will not increase along with the received acknowledgements, but keep constant to avoid the congestion by reducing the data transmission rate. If the network status changes better, the coordinate returns to the above, then congestion notice is relieved, and continues to perform the original TCP congestion control algorithm.

Simulations

We continue to use the network topology shown in Fig. 1. A sends TCP data to B. C sends UDP data to D, and the UDP data rate reaches 2.4Mbps, which has already exceeded the capacity between

these two routers, so congestion occurs. The simulation time is 100 seconds and the original TCP algorithm is still used. Because the UDP data rate is too high, much more packets reach the buffer of router R1, TCP packets reaching the buffer are discarded with high possibility. The TCP congestion window of the simulation results is shown in Fig. 7. The congestion collapse occurs at least 3 times, which results in the sharp decrease in the congestion window as well as the TCP throughput.

Next we apply the decision boundary of the linear neural network to TCP protocol. The TCP congestion window is shown in Fig. 8. We can find the congestion state judged by the decision boundary is still going on, so the TCP congestion window doesn't continue to increase with the coming of ACKs in the congestion avoidance stage, but keeps stable. The congestion collapse is not found, meanwhile the TCP throughput increases slightly.

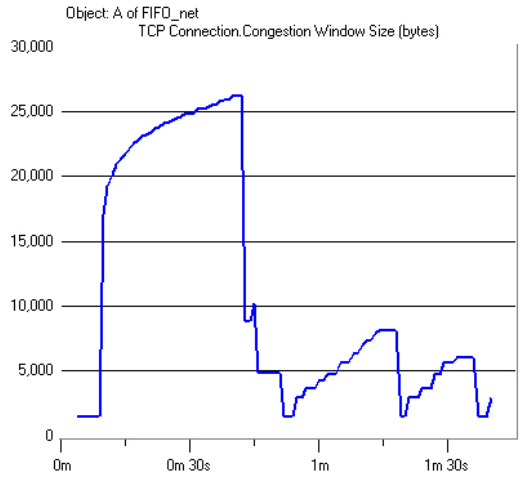

Fig. 7 The CWND of original algorithm

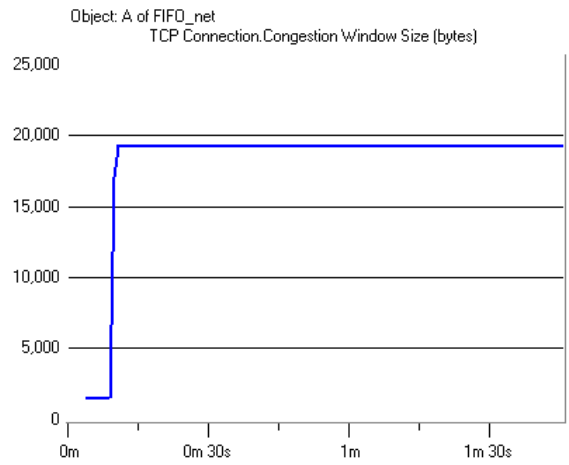

Fig. 8 The CWND by using the decision boundary

Summary

According to the variation of the RTT and the throughput of the TCP connection in the congestion and non-congestion state, we input them into the linear neural network and train the network with samples and get a decision boundary. We apply the decision boundary to TCP protocol. The simulation results show that it can effectively predict the congestion and reduce the probability of congestion collapse.

Acknowledgment

This work is supported by the fund of scientific research of education department of Anhui province (2015KJ014), and the fund of National Statistical Science Research (2014LZ32).

References

[1] Jacobson V, Congestion avoidance and control, ACM Comput. Commun. Rev. 18(1988) 314-329.

[2] A. B. M. Alim Al Islam, V. Raghunathan, iTCP: an intelligent TCP with neural network based end-to-end congestion control for ad-hoc multi-hop wireless mesh networks, Wirel. Netw. 21 (2015) 581-610.

[3] Chi-Sen Li, Mu-Chen Chen, Identifying important variables for predicting travel time of freeway with non-recurrent congestion with neural networks, Neural Comput. Appl. 23(2013)1611-1629.

[4] R. L. Ujjwal, C. S. Rai, and Nupur Prakash, Adaptive Congestion Controller for ABR Traffic in ATM Network, Wireless Pers. Commun. 70(2013)759-768.

[5] N. Xiong, L. T. Yang, Y. Yang, J. H. Park, and G. Wei, Design of QoS in Intelligent Communication Environments Based on Neural Network, Wireless Pers. Commun. 56(2011) 97-115.

[6] A. Kesselman, Y. Mansour, Adaptive AIMD Congestion Control, Algorithmica. 43(2005)97-111.