Dynamic Vegas: A Competitive Congestion Control Strategy

Keren Zhou, Qian Yu, Zhenwei Zhu and Wenjia Liu

Abstract TCP Vegas is regarded as balanced congestion control strategy, however, shortcomings such as low bandwidth utilization and not gain fairness when sharing link with TCP Reno. We reinvestigate these issues and propose a modification strategy called Dynamic Vegas, which changes initial TCP Vegas' in slow start and congestion avoidance phase. It dynamically chooses slow start algorithm and adjust decrease/increase rate in congestion avoidance phase according to specific network environment. Experiments show that within a single link network, it performs as well as TCP Vegas. Additionally, in Multi-link environment, it achieves strong competitiveness against TCP Reno and gains fairness in throughput of all senders in the end of transmission.

Keywords Slow start · Congestion avoidance · Competitiveness · Fairness

1 Introduction

With the development of network communication technology, there are mainly 4 versions of TCP protocol: TCP Tahoe, TCP Reno, TCP NewReno and TCP SACK.

TCP Tahoe which is the earliest version includes three basic algorithms: "slow start", "congestion avoidance" and "fast retransmit". TCP Reno is added the "fast recovery" on the basis of TCP Tahoe. TCP NewReno modifies "fast recovery" process by taking a situation that a large quantity of data packages was lost in a

K. Zhou (✉) · Q. Yu · Z. Zhu · W. Liu
School of Software, Yunnan University, Kunming 650091, China
e-mail: robinho364@gmail.com

Q. Yu
Key Laboratory in Software Engineering of Yunnan Province, Kunming, China

S. Patnaik and X. Li (eds.), *Proceedings of International Conference on Computer Science and Information Technology*, Advances in Intelligent Systems and Computing 255, DOI: 10.1007/978-81-322-1759-6_40,

send window. TCP SACK which modifies the acknowledgement mechanism only retransmits the data packages lost.

A new congestion control strategy-TCP Vegas is invented by [1]. TCP Vegas detects whether network is congested or not by testing RTT to change the size of congestion window. For example, Vegas decreases the size of congestion window when detects that network is congested when RTT is becoming larger. Conversely, if RTT is becoming smaller, Vegas increases the size of congestion window.

The biggest advantage of TCP Vegas is the congestion mechanism which relies on RTT. It provides a more accurate way to predict available bandwidth and let TCP Vegas to be a fair and efficient protocol.

However, TCP Vegas cannot be used in large scale network, because it could not compete with other strategies such as TCP Reno [2]. The reason is that routers in network buffer data packages will lead the increment of RTT, whereas, it does not always mean that the network is congested. Thus, transmission delay will increase, RTT cannot provide an accurate measurement of the transmission rate. Back to the original TCP Vegas mechanism, to avoid congestion, the congestion window decreases. In the end, it can share fairness with TCP Reno. This situation is mostly happened in the wireless network.

Our study which can be divided into slow start phase and congestion avoid phaseis aiming at improving the competitiveness of TCP Vegas. In slow start phase, our strategy dynamically choosing slow start algorithm. In congestion avoid phase, we adjust the congestion window by the ratio of differs about predictive size and real size to improve TCP Vegas's efficiency.

2 Issues With TCP Vegas

TCP Vegas maintains two values: *ActualRTT* and *BaseRTT*, and values are changed each time receiving a valid ACK.

$$Expected = \frac{Cwnd}{BaseRTT} \tag{1}$$

$$Actual = \frac{Cwnd}{ActualRTT} \tag{2}$$

$$Diff = Expected - Actual \tag{3}$$

Expected is the expected transmission speed, *Actual* is the actual transmission speed. *Diff* is the diff of expected speed and actual speed.

In the congestion phase, Vegas defines two constant α and β. The size of the congestion window is increased by 1 when $Diff < \alpha$; when $Diff > \beta$, it decreases by 1. Otherwise, it remains unchanged.

$$Cwnd = \begin{cases} Cwnd + 1 & (Diff < \alpha) \\ Cwnd - 1 & (Diff > \beta) \\ Cwnd & (\alpha > Diff > \beta) \end{cases} \quad (4)$$

This algorithm will adjust the value of *Diff* to be $\alpha > Diff > \beta$, which indicates that resource of the network is balanced. And for the reason that α and β are set manually, it could flexibly fit specific network environment.

Although TCP Vegas can provide better performance than other protocols, and it has less jitter and retransmission, there are still many drawbacks about TCP Vegas, such as fairness, rerouting, and unfair treatment of connections [3].

Fairness is concerned when TCP Vegas and TCP Reno are both in the network, TCP Reno's *cwnd* is increasing consistently until a packet is lost. However, it leads to the increment of actual RTT measured by TCP Vegas, which results in the decrement of *cwnd* of TCP Vegas.

Rerouting is about traditional TCP Vegas performs badly when transmission route is changed. A method [4] called "active spurring" is provided to solve the problem, which has a basic idea that break down the balance when *cwnd* is stabilized.

3 Dynamic TCP Vegas

We present our improvement of TCP Vegas in this section. The core of our modification is dynamically choosing a strategy to fit the bandwidth.

In slow start phase, our algorithm detects the bandwidth of the route to decide which strategy be used.

In congestion control phase, the original TCP Vegas algorithm adjusts the value of *cwnd* by a constant number, no matter how much the ratio of expected transmit speed to current transmit speed.

In our modified algorithm, *cwnd* is dynamically changed according to current congestion degree. The recovery algorithm of dynamic Vegas is the same as that of Vegas. We discuss details of our adjustment in the following part.

3.1 Slow Start

As mentioned before, the slow start strategy is to dynamically choose algorithm. Additionally, a new algorithm called "mid speed start" is introduced to promote the performance of dynamic TCP Vegas. To begin with, we detect the bandwidth of the transmission route. Two variable delta and alpha in TCP Vegas is used to separate the situation into three categories as follow:

If delta is less than alpha, it indicates that utilization of network bandwidth is low. Thus, we apply "mid speed start" to quickly utilize the bandwidth.

The "mid speed start" algorithm is divided into following three stages:

The first stage. Set the value of *ssthresh*, and set the *cwnd* with half of *ssthresh*. The initial increase value of *cwnd* is *cwnd*/2.

The second stage. The value of *cwnd* increases with half of its increasing value in previous RTT.

The third stage. *cwnd* increases by one each RTT.

Intuitively, it is a reverse of the traditional slow start algorithm in TCP Reno which begins at a low speed, and goes exponentially. Thus, when both dynamic Vegas and TCP Reno are applied in the network, our algorithm has strong competitiveness.

If *delta* is greater than *alpha* and less than *beta*, we use traditionally start strategy in TCP Reno to utilize bandwidth.

If *delta* is greater than *beta*. Obviously, the transmission route is congested. The slow start strategy in Dynamic Vegas increases*cwnd* every two RTT.

Algorithm SlowStart()

```
1:  If receive first ack in slowstart then //we determine slow-start algorithm when first ack arrives
2:      if delta<alpha then//indicate that bandwidth is sufficient
3:          cwnd=ssthresh/2
4:          flag=1
5:      else if alpha<delta<beta then//indicate that bandwidth is as we suppose
6:          flag=2
7:      else//bandwidth is to some extent congested
8:          flag=3
9:  else if cwnd<ssthresh then//applying dynamic vegas
10:     if flag=1 then
11:         if cwnd/2>1 then//mid-speed start
12:             incr=cwnd/2
13:         else
14:             incr=1
15:     else if flag=2 then//common start
16:         incr=1
17:     else//vegas start
18:         incr add 1 for every other RTT
```

3.2 Congestion Control

Recall TCP Vegas's strategy which adjusts the sending rate to keep network transmission steady. Weakness in that strategy is modifying the value of *cwnd* by a constant number so that the speed is changed slowly. Thus, a more adaptive idea is to modify *cwnd* dynamically by *delta* and *alpha*.

$$ratio = \begin{cases} \frac{delta-beta}{beta}, & delta > beta \\ \frac{alpha-delta}{alpha}, & delta < alpha \end{cases} \tag{5}$$

If *delta* is greater than *beta*, which means network bandwidth is congested, the greater value of *delta*, the quicker ratio is decreasing. If *delta* is less than *alpha*, which indicates that network bandwidth is not utilized, sending rate should be increased. The closer *delta* to zero, the higher the *ratio*. However, since we are in the congestion control phase, the increasing speed could not be too high. If *delta* is greater than *alpha* and less than *beta*, which reflects that the actual sending rate is close to current sending rate, thus, *cwnd* remains unchanged.

Algorithm Congestion()

```
1:   If delta>beta then
2:      ratio=(delta-beta)/beta
3:      incr=-ratio//decrease cwnd by ratio
4:      if cwnd<2 then
5:         cwnd=2
6:   else if delta<alpha then
7:      ratio=(alpha-delta)/alpha
8:      incr=ratio//increase cwnd by ratio
9:   else
10:      in
```

4 Simulation

In this section, we set up two simulations by using NS-2 software.

The first one is a single link TCP simulation. Throughput between traditional TCP Vegas and Dynamic Vegas are compared.

The second one is a multiple TCP connection simulation which consists of several receivers and senders. We have compared attributes such as *throughput* and *cwnd* of TCP Vegas and Dynamic Vegas in both heterogeneous and homogeneous network.

Table 1 The comparison between vegas and dynamic vegas's throughput capacity

Properties	Vegas	Dynamic Vegas
Throughput	3948.0 kbps	3933.6 kbps

4.1 Single TCP Connection Simulation

First we test the performance of Vegas and Dynamic Vegas in single high bandwidth-delay connection environment, comparing their time interval from slow start phase to congestion avoidance phase and their throughput capacity.

Link sender to router1 and router2 to receiver has a bandwidth of 100 Mbps with initial RTT of 1 ms, and router1 to router2 has a bandwidth has a bandwidth of 1 Mbps with initial RTT of 40 ms. Assuming that window size is 1,024, routers' cache size are 50. The simulation runs for 40 s so as to make the connection time believable and stable. Table 1 shows the comparison of the average performance between Vegas and Dynamic Vegas.

In the condition of single link, the performance of the Vegas and Dynamic Vegas are basically identical.

4.2 Multiple TCP Connection Simulation

This simulation will compare and analyze the bottleneck link's throughput when Vegas and Dynamic are in the network simultaneously. Link bandwidth for sender to router1 and router2 to receiver are 10 Mbps, the propagation delay is 1 ms. Set router1 to router2 link bandwidth 1 Mbps, the delay as 48 ms. Assuming window size to be 1,024, router cache to be 50. Running time for this simulation is 200 s, specific topology is shown in Fig. 1 (Table 2).

Vegas tends to finish the slow start phase which has an exponential growth of congestion window as soon as possible, and then enters the congestion avoidance phase until the congestion window into balance. Because of the congestion

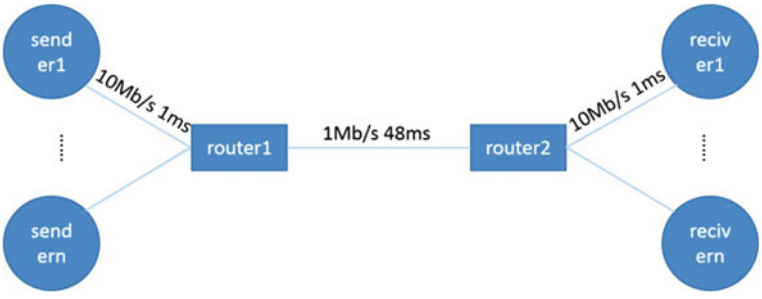

Fig. 1 Multiple TCP connections topology

Table 2 The compassion between Vegas and Dynamic Vegas when reach steady state in different networks

Properties	Vegas	Dynamic Vegas
Cwnd (Homo)	10 packets	20 packets
Cwnd (Hete)	5 packets	25 packets
Throughput (Homo)	500 kbps	500 kbps
Throughput (Hete)	200 kbps	500 kbps

window has a linear growth in congestion avoidance phase, so it takes a lot of time of Vegas to make congestion window tends to the balance. Vegas has a lot of problems such as reroute, continue to congestion and so on. And a heterogeneous network composed of Vegas and TCP Reno exists compatibility issues and causes the decrease in network's throughput which composed of Vegas and Reno.

Table 2 shows performance of Dynamic Vegas compare with Vegas in both heterogeneous and homogenous network. We find that TCP Vegas and TCP Dynamic Vegas have almost consistent performance in the single TCP connection simulation. In multiple TCP connection simulation, both of them have almost same throughput in homogeneous network. But Dynamic Vegas has a lager congestion window than Vegas, so that it will have advantages of throughput in later period. The performance of TCP Dynamic Vegas are significantly better than TCP Vegas in heterogeneous network, which reflected in the size of the congestion window and the competitiveness of the throughput. And we can realize Dynamic Vegas' great stability of congestion window phase in heterogeneous network. Dynamic Vegas' congestion window does not make too big change when Reno joining this network, which improves TCP Vegas in competitiveness and stability.

We also find some problems in our algorithm. For example, it will lead to some packets lossbecause "mid-speed-start" has high bandwidth in start phase. But we do not consider that serious defects. Because Reno also has the same problem in its start phase. We will solve this problem for further improvement in our future research.

5 Conclusion

In this paper, we come up with a modified version of TCP Vegas, TCP Dynamic Vegas, which is better than TCP Vegas in some conditions. For example, it have advantages of achieving higher throughput in multiple connection simulation. Moreover, TCP Dynamic Vegas has gained greater competitiveness than TCP Vegas in heterogeneous network. And Dynamic Vegas' congestion window does not make significant change when Reno joins in the network, which improves TCP Vegas' stability and fairness. But, there are still some problems in this algorithm, and we will solve these problems for further improvement in our future research.

Acknowledgments This work is supported by Scientific Research Fund of Yunnan Provincial Department of Education under Grant No. 2012C108; Education Innovation Fund of Software School of Yunnan University under Grant No. 2010EI13 and No. 2010EI14.

References

1. Brakmo, L.S., Peterson, L.L.: TCP Vegas: End to end congestion avoidance on a global Internet. IEEE J. Sel. Areas Commun. **13**(8), 1465–1480 (1995)
2. Mo, J., La, R.J., Anantharam, V., Walrand, J: Analysis and comparison of TCP Reno and Vegas. IEEE proceedings of eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM'99, vol. 3 (1999)
3. Srijith, K.N., Lillykutty, J., Ananda, A.L.: TCP Vegas-A: solving the fairness and rerouting issues of TCP Vegas. Conference Proceedings of the IEEE International Performance, Computing, and Communications Conference 2003
4. Zeng, Y., Wei-Jun, JZ: A rerouting issue with TCP Vegas and its solution. Comput. Sci. 8(2006), 009 (2006)