Research of Self-Adaptive of TCP Vegas Base on Dynamic Time Delay

Yueqiu JIANG, Chunqiang YANG, Qixue GUAN
Shenyang Ligong University, Shenyang, 110000, China
Tel.: (024)24686105
E-mail: missjiangyueqiu@sina.com, buyinos@163.com, guanqixue@126.com

Received: 11 July 2013 / Accepted: 25 September 2013 / Published: 31 October 2013

Abstract: TCP Vegas is a traditional congestion control algorithm based on measuring round-trip time. The performance of this algorithm is superior to the traditional TCP protocol in many respects. TCP Vegas congestion avoidance mechanisms also have some problems through the study, including competition with Reno whose aggressive performance lower (including the lower aggressive performance in competition with Reno). In order to solve these problems, this paper proposes an improved TCP Vegas congestion avoidance algorithm, TCP Vegas-E. This algorithm dynamically adjusts the alpha and beta thresholds through different network delays to approach the largest number of packet, and to make full use of the available bandwidth on the Internet. The simulation shows that the new congestion avoidance algorithm can get good performance in wired networks to solve the problem that the Vegas – E algorithm is able to enhance the bandwidth competitive ability with Reno in coexistence network environment by the NS-2 simulation software.

Keywords: TCP Vegas, Congestion avoidance, Link bandwidth, Self-adaptive.

1. Introduction

Along with quick development of the Internet, the bandwidth of the cable network cannot meet the needs of more customers. Consequently, mass of data in the physical channel leads to serious congestion problems that how to send to the network information content less than the capacity of network information and avoiding information losing had been concerned by more people. In various versions of TCP protocol, TCP Vegas (For a detailed description see [1]) is considered that its efficiency is higher, but compared with other protocols in shared network environment, its special congestion avoidance mechanisms cause the existence of some problems. The main result reported in [2] paper that Vegas is able to achieve a range from 37 to 71 % better throughput than Reno, but this improvement in throughput is not achieved a competitive retransmission strategy that effectively steals available bandwidth away from TCP connections that use the current algorithms. The issues of the TCP Vegas also are analyzed by [3]. The Vegas, however, also has some advantages, such as stability [5, 6], fairness and decreased by 20 % – 50 % packet loss rate [1, 7], etc. However, when TCP Reno compete network bandwidth with the TCP Vegas, Vegas protocol throughput is much lower than Reno protocol [12], owing to the two algorithm adopted different congestion control mechanisms [17]. Reno adopts more aggressive mechanisms to steal Vegas network bandwidth [4], yet Vegas adopts more conservative congestion avoidance mechanisms to make the algorithm respond to condition of the network early, whose mechanisms which greatly affects the performance of Vegas lead to the following two situations that the slow–start early
ends and comes into the congestion avoidance stage. Therefore, these problems have a great influence on the use of Vegas, and how to improve the competence of Vegas algorithm in the shared channel by the simulation experiment has become an important research subject.

This paper mainly studies the drawback that TCP Vegas exists [4] and the characteristics of TCP Reno own, and puts forward an improved congestion avoidance algorithm – TCP Vegas-E (Enhance) algorithm based on Vegas. According to RTT changes in network, this algorithm can automatically adjust the threshold to a reasonable size, which determines whether the sender can make fully use of the available bandwidth to approaches to the largest number of network packets, and enhanced TCP Vegas competitive power and fairness [14, 18] with Reno in the coexistent network environment.

2. TCP Vegas Protocol

In 1995 Brakmo and Peterson put forward an algorithm which is a new design for TCP that was introduced by Brakmo et al. [1, 8], according to a modified retransmission strategy based on fine-grained measurements of the round-trip time (RTT). This algorithm, Vegas, is very different from other derivative versions of TCP Reno. Because the algorithm analyzes the difference between the actual rate and expected throughput rates and compares with the $\alpha$ and $\beta$ thresholds and predicts the time when congestion will happen. The improvements are mainly concentrated in the following three aspects: New Retransmission Mechanism, New Congestion avoidance mechanism, Modified Slow-start.

Vegas algorithm differs from TCP Reno which is respectively described in detail as follows:

1) New Retransmission Mechanism: there are three strategies which affect TCP protocol retransmission. To calculate the RTT value when every segment sent by fine-grained clock values. When a duplicate ACK (acknowledgement) is received, the algorithm of TCP Vegas to judge whether the timeout period has expired. If the timeout has happened, the lost segment is going to be retransmitted. Second, when anon-duplicate ACK that is the first or second after a fast retransmission is received, TCP Vegas again checks for the expiration of the timer and may retransmit another segment [10]. Third, in case of multiple segment loss (n is set by 3) during one fast retransmission, the Cwnd is reduced.

2) Congestion avoidance mechanism: the exponential growth of the window is stop until the increasing congestion window reaches the threshold window. It will computing the measured throughput and expected throughput during the congestion avoidance stage. So adjust the congestion window by the (1) (2) (3) condition below.

3) Modified slow-start mechanism: In order to have available comparisons of the expected and the actual throughput, TCP Vegas allows to grow only every other RTT (every two RTT).

The measures are going to improve performance of TCP Vegas [19, 20].

TCP Vegas Protocol traces the relevant changes of the RTT by new congestion avoidance mechanism and adjusts its window size by (1) (2) (3).

\[
\text{Diff} = \text{Exp} - \text{Act} = \left( \frac{\text{WSize}}{\text{BaseRTT}} - \frac{\text{Traned}}{\text{ActRTT}} \right) \quad (1)
\]

\[
\text{BaseRTT} = \text{The smallest RTT} \quad (2)
\]

\[
\text{Cwnd} = \begin{cases} 
\text{Diff} < \alpha, \text{ linearly increases} \\
\text{Diff} > \beta, \text{ linearly decreases} \\
\alpha < \text{Diff} < \beta, \text{ unchanged}
\end{cases} \quad (3)
\]

Equation (2) gives RTT value which is set to BaseRTT is the minimum of all measured round-trip times, and then (1) gains the Diff value which is figured out by letting Diff=Exp−Act, where the Exp is expected throughput rate and the Act is actual throughput. The (3) works out the position of Diff value around the interval $\alpha$-$\beta$ scope, and estimates the packet number in link, thereby adjusts the Cwnd (Congestion Window) size accordingly. ActRTT represents the value of the current measured RTT and Traned indicates the ACK number during sending the packets until it receives a number of responses to ACK during a segment RTT. This slow-start mechanism is more accurate adjustment the change rate of Cwnd. So as to exactly estimate the network available bandwidth and decreases the loses, Cwnd of Vegas will double only in two round-trip times.

The behavior of TCP Vegas when the connections make full use of the available bandwidth more than threshold value is as shown in Fig. 1. This is the original algorithm of Vegas which depicts the congestion avoidance mechanism (CAM) in this figure.

There are detailed graph keyed to the following explanation:

1) When congestion happened, Vegas comes into CAM, which will judge whether Diff>β is false or true. If the condition is true, it will turn to 3.

2) That the condition 1 is false triggers this situation. And unceasingly judge the region scope of Diff.

3) TCP Vegas increases the congestion window linearly during the next RTT.

4) Vegas dose not update the congestion window linearly during the next RTT.

5) Vegas increases a little congestion window linearly during the next RTT.

There exist many better performances of TCP Vegas which shown an increasing throughput, reduced jitter and much reduced retransmissions on the Internet. But when the algorithm is connected with other protocols, TCP Vegas also produces some problem. We will describe them as follows.
1) **Rerouting problem**: the parameter baseRTT which indicates the smallest round-trip delay is used to measure the expected throughput. If the route is changed, the RTT may change. Vegas changes the congestion window size based on RTT. If the rerouting problem leads to a longer RTT, TCP Vegas presumes that because of network congestion leading to the increasing RTT. In order to solve the issue, TCP Vegas guarantees the number of packets in the extra buffer of the routers by limiting the number of packets in the router between $\alpha$ and $\beta$, when the propagation delay increases. But the solution is only affected the extra buffer in the link, it can not keep the steady congestion window size.

2) **Fairness problem**: the conservative algorithm of TCP Vegas increases/decreases the congestion window by comparing the position in the interval of the $\alpha$ and $\beta$. But Vegas has to try to maintain a smaller window size when comparing bandwidth with TCP Reno which increases the window size until a packet loss is detected. So the Reno algorithm steals a larger available network resource than the Vegas because of the Vegas shortcoming of conservative estimation. The smaller window size of the Vegas is due to the undersize $\beta$, which retransmit the router buffer size and keep a steady transmission rate in order to decrease the loss happen. So the larger router buffer size, the worse the fairness between Reno and Vegas connections [21].

3) **Slow-start threshold**: the threshold determines the time in which the algorithm enters congestion avoidance stage ends the exponential increase of the window size. If the threshold is set too higher, TCP Vegas produces an issue that TCP Vegas sends a mass of data which exceeds the capability of the link. If the threshold is set too lower, it will make the TCP turn into Slow-start stage earlier, which will not make fully use of the bandwidth in the link.

4) **The average RTT**: these are many connections on the network and the different hosts lie in various room. So the hosts produce different average RTT from the different connections, which makes the shorter connection steal more available network bandwidth. The phenomenon above directly affects the transmission rate of the long time delay link. This different RTT makes the protocol unfair network competition in unlike position hosts.

As pointed out the problems above. The algorithm exists inherently advantages and disadvantages. However Reno algorithm also has its limits, such as it detects and controls the congestion window size by the loss the packets and it will send new segment until it receives the loss data segment. We will put forward a novel congestion mechanism to fairly compete the available bandwidth with the Reno based on the feature of lose packets in the network.

3. TCP Vegas-E protocol

TCP Vegas protocol has better performance than other algorithms when no connection is occupying the network. The algorithm is responsible for the performance improvements noted in [8]. Vegas algorithm has predicted the network congestion before the actual loss is going to happen, thus reduce the window size in advance. Therefore, this algorithm is relatively weak competitiveness when both protocols shared a channel, and the limitation of TCP Vegas can not led to a wide range of applications in the wired network.

According to the properties of the TCP Vegas protocol, the new algorithm, Vegas-E which can dynamically adapt network traffic changes is put forward. Vegas-E modifies the shortcomings of congestion avoidance as a result of conservatively estimating network bandwidth. This algorithm uses the $\alpha$-$\beta$ region to estimate network bandwidth and accurately approaches to accommodate the packet number of the network, this method is perhaps more accurate to measure how many extra buffers the connection occupies on the network and calculating the difference of the latest two rates can realize the network utilization rate. Vegas-E shows high competitiveness under the coexistence network environment condition.

Vegas-E algorithm uses a technology of TCP to detect network based on measuring RTT. But the
Reno’s congestion detection and control mechanisms use the loss of segments as a signal that network is congested by data [9], during the congestion avoidance stage. Reno algorithm always detects network bandwidth, which has a great influence on the stability of the network bandwidth. TCP Reno exists itself bottleneck, so the algorithm generates a large number of segment connections going through a bottleneck with a small buffer size, leading to a large number of data lost on the network. Ait-Hellal [11] evaluate the pre-release version of TCP Vegas and Reno.

This algorithm puts $\alpha = 1$ and $\beta = 3$ [13] as the most conservative threshold values and calculates the difference between the Expected sending rate and the Actual sending to estimate the number of packets of the network, then adjusts the $\alpha$-$\beta$ thresholds region by the difference. The $\alpha$-$\beta$ threshold region is used during the next RTT. Adjusting the method is showed in the Fig. 2.

![Fig. 2. $\alpha$-$\beta$ thresholds adjustment.](image)

The algorithm compares the measured throughput rate ($Th(t)$) with the last measured throughput rate ($Th(t-RTT)$) at the congestion avoidance stage, to dynamically realize the situation of the network resources, and then appropriately adjust congestion window size by increasing or decreasing.

We describe the congestion avoidance algorithm in detail at present. Fig. 2 shows that when the difference between the expected throughput and the actual throughput is higher than the beta. Comparing $Th(t)$ with $Th(t-RTT)$ estimates the actual utilization rate of the network. If $Th(t)$ is higher than $Th(t-rtt)$ value, it indicates that the network still has available bandwidth, using the average method to add the one-half distance of between $\alpha$-$\beta$ threshold value and the Diff value makes $\beta$ closer to the biggest extra buffer in the network. The CAM ultimately triggers the changes of window size.

Using the above method adjusts $\alpha$-$\beta$ thresholds will create oscillations from left to right, and affect congestion window size, thereby regulate the number of occupying network extra buffers.

Vegas-E algorithm in the simulation process shows good stability and aggression, and there is a specific algorithm about how to modify congestion avoidance mechanisms which are described below:

\[
X = (\alpha - \text{Diff})/2.0 \quad (4)
\]

\[
Y = (\beta - \text{Diff})/2.0 \quad (5)
\]

Fig. 5 shows the behavior of TCP Vegas-E congestion avoidance mechanism. The numbers in this figure are keyed to the following explanations:

1) When congestion happened, Vegas comes into CAM, which will judge whether $\text{Diff}>\beta$ is false or true. If the condition is true, it will turn to 2.

2) Comparing the current Actual sending rate with the last measured throughput rate.

3) If condition (1) is false, it will turn to here, and continues to confirm the region of Diff.

4) And 5) indicate that different sending rate creates oscillations to occur from left to right.

5) And 6) shows that the different sending rates adjust the window size accordingly.

9) and 10) shows that different $\alpha$-$\beta$ thresholds region produces significant improvements on $\alpha$-$\beta$ region which provides a damping effect.

Equation (4) and the (5) indicate the average distance offset, which give the X and the Y computing method.

4. The Analysis of Simulation

This paper adopts network topology of Fig. 3. By using NS-2 simulation software [15]. Host 1S sending data to Host 2D and Host 2S sending data to Host 2D, which have the same bottleneck link, and the link bandwidth between the router 1 and router 2 is 1 Mbps, the router queue size is ten-each packet is 1KB and the queuing discipline is Droptail.

![Fig. 3. Network configuration for simulations.](image)

4.1. Vegas and Vegas-E Algorithm

Simulation Experiment with no Other Traffic Sources

In this experiment, the host 2S doesn’t send any data and let host 1S respectively simulates via TCP Vegas and TCP Vegas-E protocol. The window of Host 1S is set to twenty-four, and the buffer size is 15. These changes of congestion window can be seen in the time interval from 0.0-1.0 s (in Fig. 4).

The both TCP trace graphs in Fig. 4 have certain features, as illustrated in this figure. The numbers in this figure are keyed to the following explanations:

1) It indicates that TCP Vegas congestion window is higher than threshold, which will trigger CAM, but we settle on the value for $\beta$ as the smallest feasible value. The Vegas can not immediately take advantage of the extra available bandwidth, because the specific congestion mechanism. If the difference (Diff) of expected throughput rate and actual throughput rate is higher than threshold, TCP Vegas...
will decrease the congestion window size, leading to hardly occupy the available bandwidth. We can see from the Fig. 4.

**Fig. 4.** Vegas and Vegas-E exclusive channel congestion window changes.

2) The Vegas-E’s threshold is exceeded at the congestion avoidance stage, and the number 3 shows that TCP Vegas-E is able to adjust congestion window (Cwnd) according to the number of available buffers in the bottleneck. It detects and controls the congestion window size based on the resources of the link.

As shown in the Fig. 5, when Vegas algorithm turns into the congestion avoidance stage, the $\alpha-\beta$ thresholds are set to so small, which leads to the Diff higher than $\beta$ and decreases the congestion window, but there are still remaining available bandwidth in the link. Vegas algorithm conservatively estimates extra buffers in the link, which leads to a problem that Vegas cannot make full use of network bandwidth. While Vegas-E algorithm, by dynamically adjusting threshold $\alpha-\beta$ values- as shown in the Fig. 6 ([A B] values represent the changes of $\alpha-\beta$ thresholds), which makes $\beta$ close to network packet number that indicates the number of available buffers in the bottleneck. By comparing the actual throughput rate with the last measured throughput rate, TCP Vegas-E will properly adjust congestion window size. In the Fig. 6, both thresholds dramatically adjust different region by existing extra buffers.

4.2. Vegas-E and Reno Simulation Analysis

Experiment 1 and experiment 2 have the same network environment, and Host 1S respectively sends data by TCP Vegas-E and TCP Vegas. Host 2S uses the TCP Reno protocol in the same network condition, and following graph depicts more specific information that different protocol has own competition for bandwidth.

We can see in Fig. 7 and Fig. 8 in the region between 0.0 and 0.5. TCP Vegas and TCP Vegas-E’s congestion window never increase faster than Reno’s, in order to decrease losses occur, whose slow-start mechanism are conservative. But the Reno increases its congestion window until it loses packets by overrunning and doesn’t send any data to occupy available bandwidth.

**Fig. 5.** Vegas-E congestion avoidance mechanism.
360

Fig. 6. Changes of Vegas-E threshold.

Fig. 7. Vegas and Reno shared channel.

Fig. 8. Vegas-E and Reno shared channel.

These differences can be seen (in the Fig. 7 and Fig. 8) in the time interval from 2.0-10.0 s. When Vegas and Reno exist in the coexistence network environment, Vegas algorithm’s competitiveness is lower than Reno algorithm significantly, which indicates that this conservative congestion avoidance mechanism has some shortcomings. Vegas-E algorithm is put forward base on TCP Vegas, which is able to fairly compete channel with Reno (the graph in the Fig. 8 shows that behavior of Vegas-E), and congestion window change of this algorithm is relatively stable. The throughput of Vegas-E and Reno algorithm tends to fairness, which is able to explain the Vegas-E bandwidth competitiveness is obviously improved base on Vegas.

4.3. Multi-Protocol Simulation
Shared Channels

We have also simulated multi-node link. The one shown in the Fig. 9, which consists of 3 traffic sources and 3 destination nodes which receive the segment data and return the ACK information.

The dash line gives the send window size of TCP Reno which increases the window size is faster than other algorithms including TCP Vegas and TCP Vegas-E. Because the Vegas and Vegas-E own the same slow-start mechanism during the congestion avoidance stage, they impact the rate of rise. If reader wants the more detail information that TCP Vegas and TCP Vegas-E rise their congestion at the incipient stages of congestion, we will give a detail graph ranging between 0 and 2.0 s in the 5 section. The graph illustrated the window change of each of these algorithms, which can show some information that the special congestion mechanism of the TCP Vegas and TCP Vegas-E makes the competitive power not as strong as TCP Reno in the Fig. 9. But it occurs the change during the interval between 1 and 10 s, owing to TCP Reno algorithm has to repair the lost segment when the receiver receives several
duplicate ACKs, the TCP Reno won’t send any segment in the link.

So the Vegas-E will steal the available bandwidth when TCP Reno turns into fast retransmit stage. We can see from the graph which shows that The Vegas-E increases the congestion window when the TCP Reno enters the timeout in the Fig. 9. The Vegas own a conservative threshold region (the α-β), which make the Vegas compete extra bandwidth advantageously when multi-protocol compete the same link. The graph indicates that TCP Vegas-E has more competitive when multi-protocol compete a bottleneck link.

5. Detailed Graph Description

To assist the reader in developing a better understanding of the graphs used throughout this paper, Fig. 10 is a trace of multi-protocol where they connect in the same bottleneck router. The Fig. 10 is the part of the Fig. 9, which is based on probability distributions obtained from traffic traces and describes the more detail information that the window change of the different protocol in the Fig. 10.

![Fig. 10. Multi-protocol simulation.](image)

The range between 0 and 0.6 s, is the congestion avoidance stage of every algorithm. The Vegas and Vegas-E own the same slow-start mechanism in the Fig. 10. At the 0.5 s, the Reno detects the losses and controls congestion window size, and then turns into fast retransmit. The range between 0.6 and 0.8 s, the Reno didn’t sent any segment until it recover all lost segment. But at around 0.85 s, the transmitting data is lost which leads the algorithm has to increase the congestion window from the slow-start stage. The rest time, the Vegas-E adjusts the congestion window size based on the available bandwidth in the link, and the Vegas rises Cwnd based on conservative threshold values.

6. Conclusions

The paper mainly studies the α-β threshold region has a great influence on the throughput rate and then put forward a novel congestion avoidance mechanism, TCP Vegas-E, which modifies the deficiency of the original algorithm to flexibly adjust to the change of network traffic based on TCP Vegas CAM. The graphs show good performance (as illustrated in Fig. 7 and Fig. 8) in this paper that TCP Vegas-E has observably improved in the bandwidth competitiveness and stability. When no other connections exist, this congestion control mechanism is able to friendly competitive bandwidth with TCP Reno by the platform of NS-2 simulation software.

Acknowledgment

We would like to express our gratitude thanks to the Communication and Network Institute of Shenyang Ligong University for their invaluable suggestions to improve this paper. We would like to thank our respected teachers and colleagues for their guidance and support as well.

References


