Intelligent Automation And Soft Computing, 2020
Vol.26,no.4,773–781
DOI: 10.32604/iasc.2020.010112

An Improved TCP Vegas Model for the Space Networks of the Bandwidth Asymmetry

Qixue Guan and Yueqiu Jiang

College of Information Science and Engineering, Shenyang Ligong University, Shenyang 110159, China
Address: No.6 Nanping Middle Road, Hunnan New District, Shenyang City, liaoning Province

ABSTRACT
It is known that congestion in the reverse direction happens in advance of the congestion in the forward direction due to the significant bandwidth asymmetry in the two directions of the space networks, especially in the satellite networks, which enables the TCP Vegas to enter the phase of the congestion avoidance blindly and reduce the throughput of the forward direction. To solve this problem, a congestion control model, TCP Vegas-DDA, which maintains the frequency of the acknowledgments in the reverse direction is proposed. The model sets the interval time between acknowledgments dynamically based on the variation of the queuing delay in the reverse direction and reduces the impact of the queuing delay on the congestion control algorithm by amending the value of the base round-trip time. Results of the simulation indicate the problem of the low throughput in the forward direction caused by the congestion in the reverse direction can be solved, and the performance of the TCP Vegas in the asymmetric links can be improved.

KEY WORDS: Acknowledgement frequency; bandwidth asymmetry；TCP Vegas；queuing delay.

1 INTRODUCTION

COMPARED with the ground network, the space network has a longer round-trip time, higher bit-error rate and asymmetric bandwidth, etc. It lowers the performance when the TCP (Reno Jacobson V, Karels M J, 1988), is originally based on the ACK timer, and is applied on the space network. Among so many improved TCP protocols, the TCP Vegas boasts lower packet loss rates and higher and more stable throughputs than those of the TCP Reno, making it a hot topic in the space network transmission protocol research (Jacobson V, 1990).

Here is the principle of the TCP Vegas congestion control algorithm. First, calculate the RTT (Round-Trip Time) using the data packet send-time and the arrival time of the corresponding ACK (Acknowledgment). Second, calculate the expected throughput and actual throughput of the link according to the value of the RTT and the CWND (congestion window). Third, estimate the actual situation of the network based on the difference between the value of the expected throughput and that of the actual throughput (Brakmo L S, O’ Malley S W, Peterson L L, 1994). At last, control the number of data packets sent into the network by adjusting the CWND size to avoid network congestion caused by the transmission link overburden or low throughput brought by the low utilization of the link (Brakmo L S, Peterson L L, 1995).

The document points out that in a link with an asymmetric bandwidth, the ACK may delay or get lost when the uplink becomes saturated and congested earlier than the downlink and then the congestion control algorithm, which depends on information fed back by the ACK will be misled to enter the congestion control phase too early. That makes the congestion control algorithm indirectly reduce the number of ACKs sent into the uplink at the expense of cutting down the number of data packets sent into the downlink at that phase to relieve the uplink congestion (Mo J, La R J, Anantharam V, et al. 1999). In the space communication link, especially the satellite communication link with the big gap in the bandwidth between the downlink and uplink, congestion is extremely easy to occur in the uplink (from ground to satellite, reverse path) when the downlink (from the satellite to the ground, forward path) is transmitting data packets at a high rate of speed. If there is other background traffic in the uplink besides the ACKs, the

CONTACT Qixue Guan gjlyy43@163.com

congestion caused by the asymmetry of the bandwidth may become even worse (Balakrishnan H, Padmanabhan V N, Katz R H, 1997). According to the principle of the TCP Vegas, the ACKs will queue in the bottleneck uplink when there is congestion in it. All those will lead to the increasing of the uplink transmission delay and measured RTT and then will mislead the TCP Vegas to impose congestion control on the downlink and thus lower the performance of the protocol (PAN Cheng-sheng, Xuan Jing-peng, WEI De-bin, et al. 2012).

At present, the promotion of the delay measurement accuracy and the CWND adjustment efficiency is the primary focus of improving the TCP Vegas for the satellite communication system (Wang Jian-feng, Huang Guo-ce, Chen Cai-qiang, et al. 2008). There are few solutions to the uplink congestion caused by the bandwidth asymmetry. You can find some relatively typical solutions in the following documents: Document analyzes the cause of the highly decreased performance of the TCP Vegas in the asymmetric link compared with the TCP Reno in detail (Gong Changqing, Zhao Zhigang, Wang Guangxing, 2006). It also calculates the queuing delay of the data packets in the downlink based on the time-stamp, and then the calculated value of the actual throughput in the TCP Vegas algorithm is revised according to the results from the former step to lower the impact of the uplink queuing delay on the TCP Vegas congestion control algorithm to some extent. But this method, calculating the downlink queuing delay using the timestamp, may introduce the time synchronization issues between the two ends and make the improvement scheme too complicated. Based on the measurement of the queuing delay of the downlink and uplink using routers, the document further lowers the impact of the uplink status on the congestion control algorithm by proposing a method of revising the actual throughput and the base RTT at the same time (YUE Peng, ZHANG Bing, LIU Zeng-Ji, et al.2006). The document presents an idea of measuring the change of the downlink queuing delay using the time-stamp and taking the results as an assistance on the downlink congestion detection to let the congestion control algorithm control the downlink congestion more accurately, rather than revising the key parameters in the original TCP Vegas algorithm (Fu C P, Chung L C, Liew S C, 2001).

Although the above method addresses the problem in which the data packet throughput decreases after introducing the congestion control algorithm when the ACK congestion is experienced in the uplink by eliminating the impact of the uplink queuing delay on the TCP Vegas algorithm. The ACK congestion in the uplink is still not being controlled directly and efficiently, making the number of the ACKs in the uplink reduce blindly. That makes the ACK frequency in the uplink not being kept at a relatively high level so that the work pace of the TCP Vegas congestion control becomes lowered and the climbing speed of the downlink data packet throughput slows down. Finally, the performance of the TCP Vegas is compromised (Fu C P, Liew S C, 2003).

Therefore, the paper proposes an end-to-end TCP Vegas congestion control model with an active control on the uplink congestion starting with the acknowledgement mechanism cooperating with the TCP Vegas. The model provides a solution to the problem of the lower downlink throughput when the congestion happens earlier on the uplink than downlink and improves the overall performance of the TCP Vegas (Chan Y C, Chan C T, Chen Y C, 2003).

2 THE TCP VEGAS-DDA (DYNAMIC DELAYED ACKNOWLEDGEMENT), AN IMPROVED TCP VEGAS MODEL

IN order to tackle the problem of the throughput decreasing when the TCP Vegas congestion control is introduced after the uplink congestion occurs, the TCP Vegas-DDA, and the congestion control model presented in this paper, imposes the congestion control on the uplink indirectly using the revised TCP Vegas congestion control mechanism and maintains a relatively high ACK frequency in the uplink with the assistance of the adaptive delayed ACK mechanism (Chan Y C, Chan C T, Chen Y C, et al. 2004).

Major amendments made on the improved model include the original TCP Vegas, and by adjusting the ACK intervals, the number of the ACKs sent into the uplink is controlled to make sure the ACKs queue in the uplink is in an appropriate manner to keep a relatively high ACK frequency. At the same time, the calculated value of the base RTT is modified according to the intervals between the adjacent ACKs to lower the impact of the early and late ACKs on the measurement of the RTT (Ho C Y, Shih C H, Chen Y C, et al.2005).

2.1 The TCP Vegas-DDA Model Structure

The model is composed of the delayed ACK mechanism controlling the ACK intervals and the TCP Vegas correction algorithm for the congestion control, as shown in Figure 1.

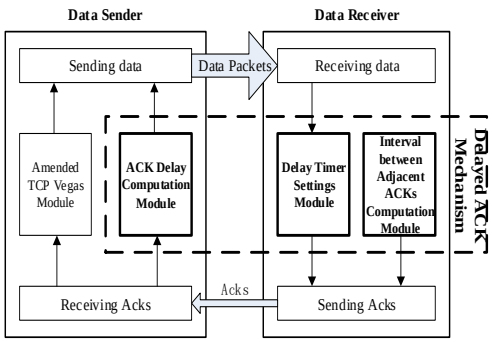

Figure 1. The Structure of the TCP Vegas-DDA Module.

The delayed ACK mechanism consists of three parts: A sender-side ACK delay computation module, a receiver-side delay timer settings module and a computation module for the actual interval between the adjacent ACKs.

In the initial stage of the model, the receiver-side ACK delay, and the maximum ACK delay is the interval between the two adjacent ACKs. Its value is set to zero. The receiver returns the ACK immediately after it receives an unacknowledged packet and there is no ACK delay(Kang Yuchi, Liu Meihong, et al, 2019).

The process of the one round of the ACK frequency adjustment is described. The sender sends data packets to the receiver. The receiver extracts the ACK delay interval, with zero as the original value, from the head of the data packet it is received and sets the ACK delay timer using the result value. Meanwhile, a decision is made whether to send the corresponding ACK at this interval according to the data packet status. Then puts the actual send-time interval between the current ACK and previous ACK into the head of the current ACK before it's sent out. After receiving the ACK, the sender computes the queuing delay of the ACK during the transmission in the uplink according to the actual send-time interval carried by the ACK and at the same time it imposes the congestion control on the uplink using the modified TCP Vegas congestion control algorithm. it calculates the ACK delay interval for the next round by taking the result queuing delay as the basis of the evaluating ACK frequency in the uplink. In the last step, it puts the result ACK interval into the head of the next data packet to be sent and a new round of adjustment begins when the data packet reaches the receiver.

2.2 Delayed ACK Strategy

On the basis of the traditional cumulative acknowledgement mechanism, the adaptive delayed ACK strategy in this model introduces a method for the delayed ACK transmission based on the ACK interval, to manage the ACK frequency in the uplink. Document points out that the delayed ACK method may lead to three problems in the traditional TCP Reno relying on the acknowledgement timer, because the ACKs received by the sender will be less or come less frequently.

(1) The burst data traffic: Assuming that the n ACKs sent by the receiver are lost or delayed due to the uplink congestion and only one ACK arrives on time finally. The slow process of the increasing CWND showing by the size of n data packets gradually during the period of accepting n ACKs has been turned into a burst that increases the CWND suddenly. By the size of the n ACKs only after receiving one ACK, the sudden expansion of the CWND raises the risk of the data packet loss. The larger the value n is, the higher the risk of the data packet loss becomes. Over-fast growing on the CWND may make the data sent into the downlink burst and the risk of data packet loss in downlink become even higher, especially in the slow-start phase.

What the TCP Vegas is based on is the change of throughput rather than the ACK arrival frequency at the sender, which means that the TCP Vegas is not a protocol based on the acknowledgement timer. Although the TCP Vegas is insensitive to the ACK arrival frequency at the sender, the calculation of the throughput still relies on the arrival of the ACKs. Therefore, in order to avoid the above problems in the slow-start phase when the CWND grows rapidly, the delayed ACK mechanism presented in this model only works when the TCP Vegas is in the congestion avoidance phase.

(2) The slow growth of the CWND: The traditional TCP Reno depends on the number of the ACKs received rather than the data volume that can be acknowledged by the ACKs to adjust the CWND. So, smaller number of the ACKs arriving at the sender successfully is bound to decrease the growth speed of the CWND.

To solve the problem, this model makes the delayed ACK mechanism work in the congestion avoidance phase when the TCP Vegas sees slow growth on the CWND to avoid any impacts on the CWND rapid growth during the slow start.

(3) The fast-retransmission malfunction: The traditional TCP Reno performs the fast-transmission as soon as it receives the three duplicate ACKs. It is possible that the sender can't receive all the three duplicate ACKs before the sender-side timer time out due to the delay of the ACK. That will prevent the sender from starting the fast retransmission when there is a data packet loss, so that the retransmission of the lost data packet can only be triggered until timeout. Therefore, performance of the protocol becomes lower.

Document [3-4] says that the TCP Vegas improves the TCP Reno retransmission mechanism. That is, the transmission time interval can be calculated using the timestamp in a duplicate ACK after receiving it with no need to get three duplicate ACKs. Then it figures out whether to start the fast-retransmission by checking the pre-set threshold of the fast-retransmission timeout. Even if there is multiple data packet loss, it can still find out in the same way, by examining the time-stamp carried by the first or second ACK (not a duplicate ACK) arrives after the successful retransmission of the first lost packet, whether to perform the fast-retransmission. Although the original TCP Vegas has already made the starting of the fast-retransmission, it no longer relies on the number of duplicate ACKs received since its born, it's still possible that the fast-retransmission cannot be started due to the lack of enough ACKs received by the sender, because the ACKs can still arrive at the

sender with a too low frequency when one of the following three situations occurs: There is a small CWND; the sender enters the slow-start phase due to timeout; and there is no packet sent out by the sender during an interval longer than an RTT. It leads to a lower performance. Therefore, an ACK time-out timer should be set at the receiver-side to ensure the ACKs arrive before time-out at the sender-side. The timer is set to 1.2 times of the RTT.

2.3 Calculation of the Interval between the Adjacent ACKs

In order to keep a relatively high ACK frequency in the uplink, an ACK queue of a certain length must be maintained in the uplink. Therefore, the interval between the adjacent ACK packets and the queuing delay of the ACK should be correlated. If an ACK is sent after an interval of T following the previous ACK, which decreases (or increases) the queuing delay 'T' of the ACK in the uplink, the maintenance on the size of the ACK queue in the uplink works. In a space communication environment with a big gap in the bandwidth between the downlink and uplink, the congestion is experienced earlier in the uplink than downlink and the packet loss caused by the error code is not taken into consideration, the primary factors making the RTT vary and is the variation in the queuing delay of the ACKs in the uplink and the variation in the ACK interval (packet delay in downlink is neglected, because there is no congestion in it), so the RTT variation should be:

$$\Delta RTT = \Delta T + \Delta T' \qquad (1)$$

In the equation: ΔRTT is RTT variation; ΔT is the variation in the ACK interval; $\Delta T'$ is the variation in queuing delay of the ACK in the uplink.

In order to keep the size of the ACK queue in the uplink and reduce the increment of the RTT as much as possible, the paper supposes the variation in the ACK and the interval is equal to the variation in the queuing delay of the ACK in the uplink, $\Delta T = \Delta T'$. Equation, $\Delta T = \Delta RTT/2$, can be derived from equation (1), therefore the ACK interval for the new round should be:

$$T_i = T_{i-1} + \Delta RTT / 2 \qquad (i=1, 2\cdots) \qquad (2)$$

In this equation, T_i is the ACK interval in the new round (with zero as its initial value); $T_{i-1}$is the ACK interval in the previous round and ΔRTT is the current RTT variation.

(1) Determination of the lower limit of the ACK interval.

The ACK interval should not be too short so that the length of the ACK queue in the uplink cannot be too long. The minimum interval of the ACK delay (lower limit of the interval) should be no less than ΔT, the ACK interval under a regular ACK sending frequency in the TCP Reno.

Δt is set as the interval between ACK_i and $ACK_{i+1,}$ the two ACKs arrive at the sender in succession.

A is set as the difference of the acknowledged data size in bytes between the two qualified normal ACKs arriving at the sender in succession. In order to get the same results as the accumulative acknowledgement strategy in the TCP Reno, this value is set to the size of 2 MSSs, which means one ACK is sent out per two data packets received.

Δa is set as the difference of the acknowledged data size in the bytes between two actual ACKs, ACK_i and $ACK_{i+1,}$ arriving at the sender in succession. After the arrival of ACK_{i+1}, the adjustment range of the data packets sent by the sender and CWND should be similar to that in the situation when $ceil(\Delta a / A) - 1$ ACKs are received, $\Delta a / \Delta t = A / \Delta T$. *Ceil* returns the smallest integer value not less than the number specified as an argument. Minus one means removal of ACK_i, the one already arrives.

The formula for calculating T_{min}, and the minimum ACK interval, is as follows:

$$T_{min} = \Delta T = (A / \Delta a) \times \Delta t \qquad (3)$$

(2) Determination of the upper limit of the ACK interval.

To ensure enough feedback is being provided while tuning the CWND with the TCP Vegas-DDA, two ACKs must be successfully received by the sender within two RTTs, of the CWND adjusting cycle. The ACK interval should be no greater than one RTT. The formula for calculating the T_{max}, the maximum ACK interval, is as follows:

$$T_{max} = RTT \qquad (4)$$

Finally, the formula for calculating the T_i, shows the ACK interval in a new round, is as follows:

$$T_i = \min(\max(T_i, T_{min}), T_{max})(i = 1, 2\ldots) \qquad (5)$$

2.4 The Revision to the TCP Vegas

The original TCP Vegas calculates the expected transmission speed and the actual transmission speed with formula's (6) and (7) respectively:

$$Expected = cwnd(t) / baseRTT \qquad (6)$$

$$Actual = cwnd(t) / RTT \qquad (7)$$

In the equation, *Expected* is the expected transmission speed and *Actual* is the actual transmission speed. *cwnd(t)* is the size of the CWND now of *t and baseRTT* is the minimum RTT in the current link observed by the sender. *RTT* is the measured RTT in the current link now of *t*.

Then work out Δ with formula (8),

$$\Delta = (Expected - Actual) \times baseRTT$$

$$= (1 - baseRTT / RTT) \times cwnd(t) \qquad (8)$$

At last, adjust the size of the CWND with formula (9) to make the number of data packets in the downlink close to the capacity of the link as much as possible.

$$cwnd = \begin{cases} cwnd + 1 & \Delta < \alpha \\ cwnd & \alpha \le \Delta \le \beta \\ cwnd - 1 & \Delta > \beta \end{cases} \quad (9)$$

It can be derived from formula (9) that the CWND adjustment algorithm varies according to the value of Δ α and β, and the measurement of Δ is related to the *cwnd* and the ratio of the *RTT* to the *baseRTT* at that moment according to formula (8). So, when α and β are set to 1 and 3 respectively, the congestion control result in the downlink depends on the ratio of the *RTT* to the *baseRTT*. Although the *RTT* includes the uplink queuing delay of the ACK, which accounts for the major part during congestion in the uplink. The value of the *RTT* is not revised by removing the uplink queuing delay while only the *baseRTT* is revised, because the TCP Vegas congestion control strategy is used in the uplink congestion control in this model.

The calculated value of the *RTT* should include the ACK delay, because the ACK is not sent out at once when a data packet arrives. The delay should be approximately equal to ΔT, (the interval between adjacent ACKs). Therefore, the revision must be performed while calculating the *baseRTT* to make its value equal to the sum of the ΔT and the previous *baseRTT*, and $baseRTT_i = baseRTT + T_i$., because *BaseRTT* and *RTT* add ΔRTT synchronously. The *BaseRTT* is usually less than the *RTT*, Δ reduces when the *cwnd* remains unchanged according to formula (8) and Δ tends to go back to the value less than α (the growth phase of the *cwnd*) according to formula (9). Therefore, the growth of the cwnd is boosted through the *BaseRTT* revision while the ACK frequency is maintained and finally the downlink throughput is improved.

3 THE SIMULATION ANALYSIS

IN order to analyze and verify the performance of the TCP Vegas-DDA model in the satellite network with the highly asymmetric link bandwidth, this paper describes the simulation of the TCP Reno, the TCP Vegas and the TCP Vegas-DDA by using the OPNET as the simulation test software. The network topology is shown in Figure 2.

Set the link between Router A and Router B as the bottleneck link. The bandwidths of the downlink and uplink are set as 1.5M/s and 1.5K/s (Bandwidth ratio is 1000:1) respectively. The one-way transmission time is 300ms; the downlink data packet size is 1000 bytes and the ACK size is 50 bytes; bottleneck downlink buffer size is 10 data packet size; duration of simulation is 20 minutes.

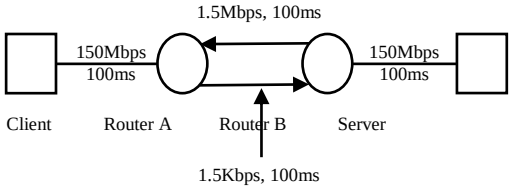

Figure 2. The Topology of the Simulation Network.

According to the definition of the standard bandwidth ratio k given in document [6], in the current simulation environment, the standard bandwidth ratio k = (downlink bandwidth/uplink bandwidth)/(data packet size/ACK size) = (1.5Mbps/1.5Kbps)/(1000 bytes/50 bytes) = 50. This means that for every k=50 data packets sent by the downlink, one corresponding ACK is sent by the uplink and both the downlink and uplink become saturated at the same time. Therefore, in the current simulation environment, if the traditional cumulative acknowledgement policy is applied, the receiver will create and send one ACK for every two data packets received. The bandwidth ratio should be no greater than 40:1 to avoid the uplink experiencing congestion earlier than the downlink. Apparently for the current simulation settings with the bandwidth ratio 1000:1, a huge amount of the ACKs would queue in the bottleneck uplink during the data transmission and they may lose when the bottleneck uplink buffer is small.

3.1 The Simulation Experiment with Unlimited Uplink Buffer

To better observe and analyze the impact of the uplink queuing delay on the TCP Vegas-DDA performance the bottleneck uplink buffer size is set to a large value. The size of 5 ACKs, and to cache the ACKs not transmitted in time. Thus, the ACKs only queue in the uplink so that the simulation will not be affected by the ACK loss.

(1) Throughput Analysis

As shown in Figure 3, the average throughputs of the TCP Vegas-DDA、TCP Vegas and TCP Reno in the downlink are 52560bit/s, 25013bit/s and 16762bit/s respectively. The TCP Vegas-DDA has an average throughput of 110% higher than that of the original TCP Vegas and the throughput of the traditional TCP Reno is the lowest. It demonstrates that the TCP Vegas and the TCP Reno can only reduce the number of the ACKs sent into the uplink indirectly at the expense of lowering the downlink throughput to relieve the uplink congestion by performing the congestion control when the uplink is experiencing congestion earlier than the downlink.

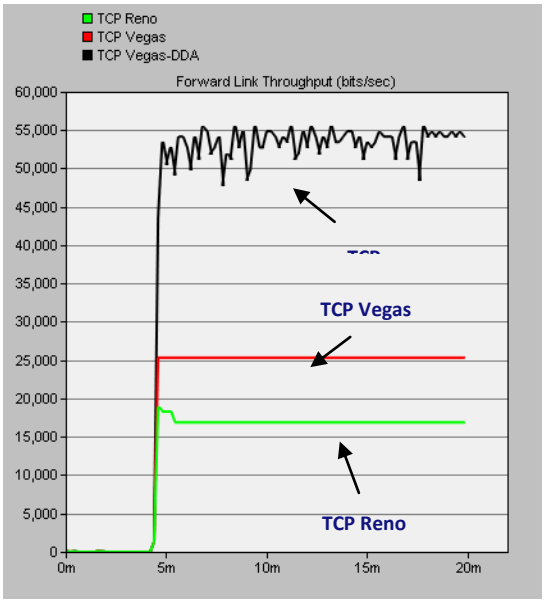

Figure 3. **Throughput in the forward direction**

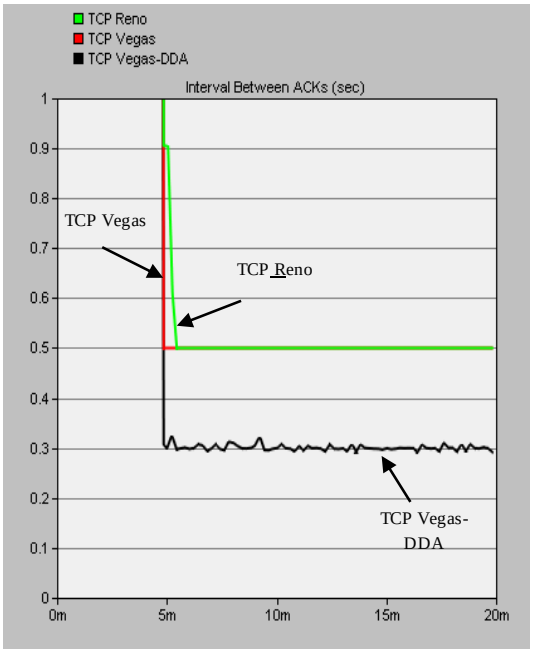

Figure 4. **The Interval Time between the ACKs.**

As shown in Figure 4, the uplink ACK interval of the TCP Vegas-DDA is always lower than those of the TCP Vegas and TCP Reno. As shown in Figure 5, the uplink throughput of the TCP Vegas-DDA is higher than those of the TCP Vegas and TCP Reno. Although the TCP Vegas-DDA may reduce the number of ACKs sent into the uplink through the congestion control mechanism when the congestion occurs in the uplink, just like the TCP Vegas. The TCP Vegas-DDA ensures a high ACK feedback frequency to accelerate the growth of the downlink packet throughput under the congestion control mechanism, because it adopts the adaptive delayed ACK strategy to help keep a certain number of ACKs in the bottleneck uplink by reducing the ACK interval. Besides that, the ACK intervals of the TCP Vegas and the TCP Reno in Figure 4 are stable and close to each other. The major reason for the volatility of the ACK queue size in the TCP Vegas-DDA is the former two protocols both use the traditional cumulative acknowledgement scheme with the static ACK frequency, making the ACK interval fixed. The TCP Vegas-DDA adopts an adaptive delayed ACK strategy, which adjusts the ACK frequency according to the queuing delay, making the ACK interval fluctuate.

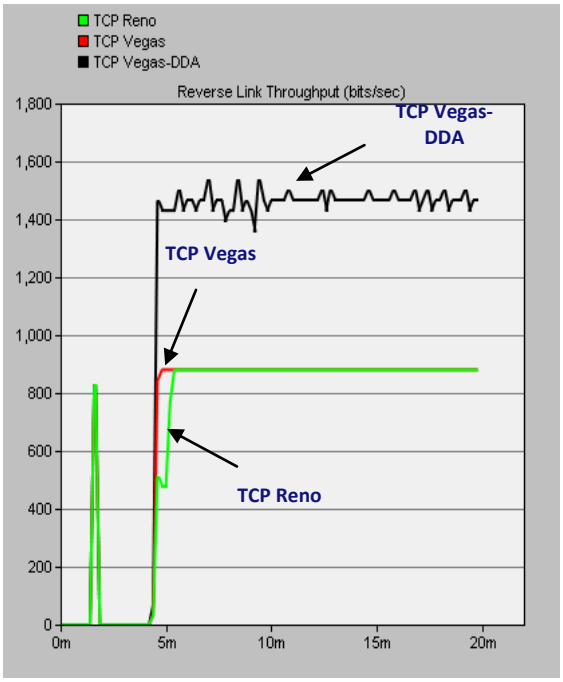

Figure 5. **The Throughput in the Reverse Direction.**

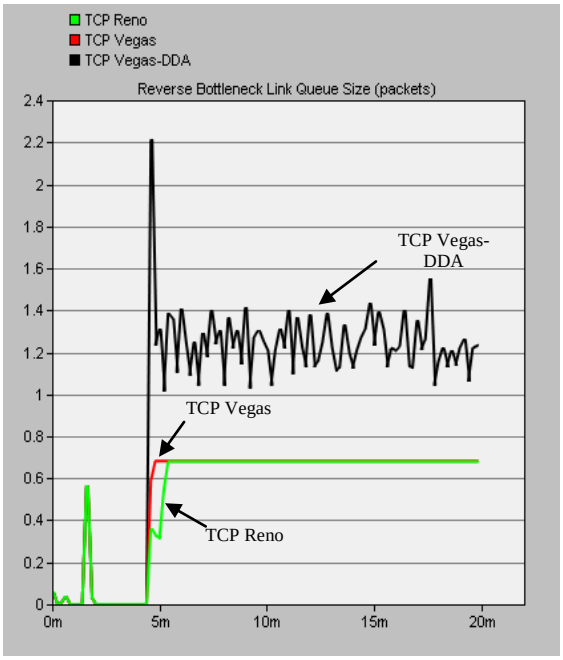

Figure 6. The Bottleneck Link Queue Size in the Reverse Direction.

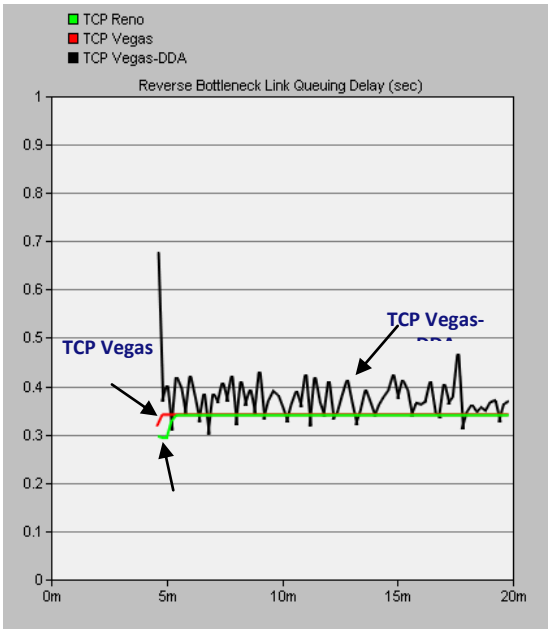

Figure 7. The Bottleneck Link Queuing Delay in the Reverse Direction.

(2) Status analysis on the queue of the bottleneck uplink.

Changes on the size of the bottleneck uplink queue are shown in Figure 6. The average length of the queue of the TCP Vegas-DDA fluctuates slightly around the size of 1.2 ACKs. Changes of the queuing delay induced by that is shown in Figure 7. The average queuing delays for the three protocols are 0.37 seconds, 0.34 seconds and 0.34 seconds, respectively. It is the involvement of the congestion control mechanism who makes the length of the queue towards a stable value. The ACK frequency under the congestion control mechanism approaches the available bandwidth of the link and then becomes stabilized, which makes the corresponding ACK generation frequency stabilized too, so that the length of the uplink ACK queue is kept stable. The adaptive delayed ACK scheme in the model causes the fluctuation of the ACK queue length in the bottleneck uplink in the TCP Vegas-DDA. When the number of ACKs queuing in the uplink increases, the ACK queuing delay rises. That makes the ACK queuing delay increase and the ACK interval grow. Consequently, the number of ACKs sent into the uplink per unit time reduces and the number of the ACKs in the queue decreases. The queuing delay may become shorter when the length of the ACK queue goes smaller, making the TCP Vegas-DDA cut down the ACK interval instead. Finally, the length of the ACK queue in the uplink begins fluctuating.

The TCP Vegas and TCP Reno have limited ability to handle the bottleneck uplink and they both use cumulative acknowledgement schemes with no adjustment on the ACK frequency. Therefore, their ACK frequency stabilizes at a fixed value under the congestion control mechanism, making the length of the ACK queue remain stable in the bottleneck uplink.

3.2 The Simulation Experiment on the Buffer-limited Uplink

As shown in Figure 6, when the buffer size of the bottleneck uplink is large, the average uplink buffer size needed by the TCP Vegas-DDA is the size of 1.2 ACKs. This experiment revises the buffer size of the bottleneck uplink to the size of 1 ACK to observe the performance of the uplink when the ACK loss happens during the congestion.

(1) The Throughput Analysis

As shown in Figure 8, when there is a small buffer size in the bottleneck uplink, the downlink throughput of the TCP Vegas-DDA is still higher than those of the TCP Vegas and TCP Reno. But the downlink throughputs of all the three algorithms, TCP Vegas-DDA, TCP Vegas and TCP Reno become higher compared with those in the situation with larger buffers and the larger throughput results from a longer bottleneck uplink ACK queue. As measured in the experiment, the ACK queuing delays for the three protocols are all 0.3 seconds, because the bottleneck uplinks are saturated and because of packet loss, intervals between adjacent ACKs arriving at the sender misleads the system to believe there is a faster pace than that with a larger bottleneck uplink buffer, it is called the Stretch ACK [17]. The ACK feedback accelerates, which bursts the data in the downlink and hence improves the throughput. The TCP Reno, which relies on the acknowledgement timer, is sensitive to

the frequency of the ACK arrival so the TCP Reno sees the largest increase in the throughput. As data packets arrive at the receiver with a higher frequency, the CWND adjustment cycle is cut down, and the work pace of the congestion control accelerates and finally performance of the protocol improves.

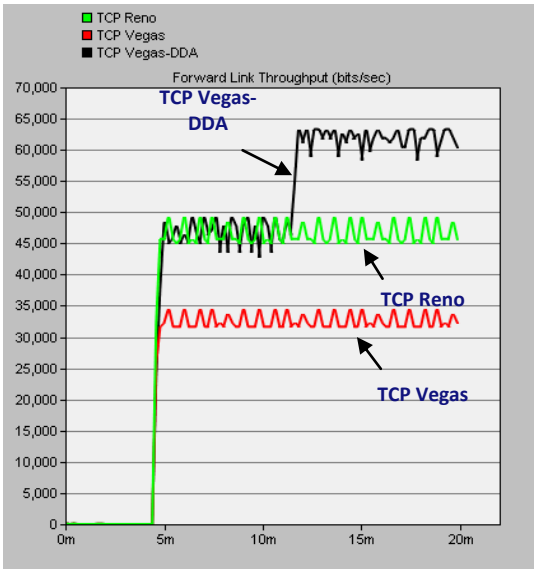

Figure 8. The Throughput in the Forward Direction.

(2) The Status Analysis on the queue of the bottleneck uplink

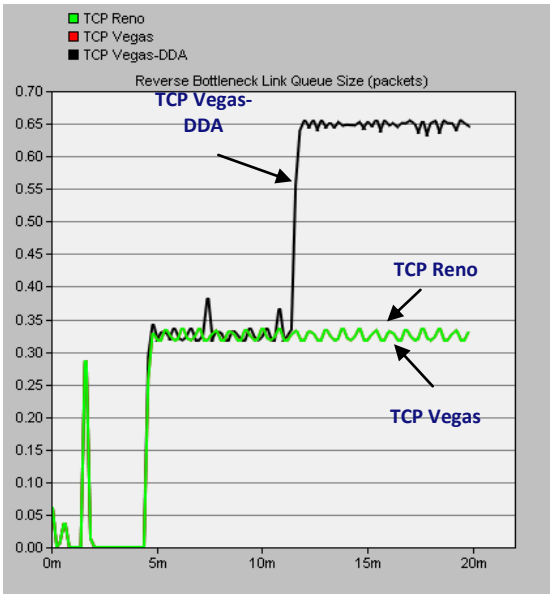

Figure 9. The Bottleneck Link Queue Size in the Reverse Direction.

As shown in Figure 9, the TCP Vegas-DDA has a more saturated ACK queue in the bottleneck uplink compared with that of the TCP Vegas and TCP Reno. It means that in a same network environment, the TCP Vegas-DDA keeps an ACK queue with a certain length in the bottleneck uplink to ensure a relatively rapid ACK feedback and then makes the congestion control work in a faster pace so that the packet throughput of the downlink can improve in a short time. As shown in Figure 10, the reduction of the ACK interval at 12 minutes is the major cause of the increment on the length of the ACK queue in Figure 9 after 12 minutes.

4 CONCLUSION

THIS paper presents the TCP Vegas-DDA, a congestion control model based on the changing uplink delay. It maintains a certain ACK feedback frequency in the asymmetric link bandwidth satellite network by dynamically adjusting the ACK interval. The model provides a solution to the problem that the downlink data packet throughput decreases after the TCP Vegas congestion control mechanism is involved when congestion occurs in the uplink. Effectiveness of the model has been verified by simulation with the OPNET.

5 ACKNOWLEDGMENT

THIS work was supported by the National Natural Science Foundation of China (61501307, 61471247, 61373159), Liaoning specially-hired Professor Program and Program for Liaoning Excellent Talents in the University (LR2015057), and it was also sponsored by the "Liaoning BaiQianWan Talents Program (2014921044)", General Project of the Liaoning Provincial Committee of Education (No.L2014078, No.L2015459), and Shenyang Ligong University (4771004kfx24).

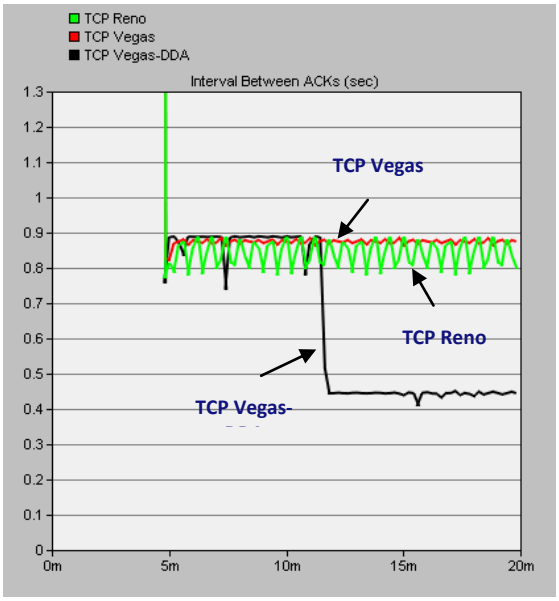

Figure 10. The Interval Time between the ACKs.

6 REFERENCES

Balakrishnan H, Padmanabhan V N, Katz R H (1997). "The effects of asymmetry on TCP performance". *The 3rd Annual ACM/IEEE International Conference on Mobile Computing and Networking, New York, USA*, 26-30.

Brakmo L S, O'Malley S W, Peterson L L (1994). "TCP Vegas: new techniques for congestion detection and avoidance". *ACM SIGCOMM Computer Communication Review*, 24(4): 24-35.

Brakmo L S, Peterson L L (1995)." TCP Vegas: end-to-end congestion avoidance on a global Internet". *IEEE Journal on Selected Areas in Communications*, 13(8): 1465-1480.

Chan Y C, Chan C T, Chen Y C, et al (2004). "Performance improvement of congestion avoidance mechanism for TCP Vegas". *The Tenth International Conference on Parallel and Distributed Systems, Newport Beach, USA*, July 9-9.

Chan Y C, Chan C T, Chen Y C. An enhanced congestion avoidance mechanism for TCP Vegas[J]. IEEE Communications Letters, 2003, 7(7): 343-345.

Fu C P, Chung L C, Liew S C (2001). "Performance degradation of TCP Vegas in asymmetric networks and its remedies". *ICC 2001. IEEE International Conference on Communications, Helsinki, Finland*, 11-14.

Fu C P, Liew S C (2003). "A remedy for performance degradation of TCP Vegas in asymmetric networks". *IEEE Communications Letters*, 7(1): 42-44.

Gong Changqing, Zhao Zhigang, Wang Guangxing (2006). "Research of TCP Congestion Control Algorithm over LEO Satellite Networks". *Computer Engineering*, 32(18): 90-92. (in Chinese)

Ho C Y, Shih C H, Chen Y C, (2005). "An aided congestion avoidance mechanism for TCP Vegas". *The Third International Conference on Networking & Mobile Computing, Heidelberg, Berlin*, August 2-4.

Jacobson V (1990). "Modified TCP congestion avoidance algorithm". *End2end-Interest Mailing List*, 16(3):265-280.

Jacobson V, Karels M J (1988). "Congestion avoidance and control". *ACM SIGCOMM Computer Communication Review*, 18(4): 314-329.

Kang Yuchi, Liu Meihong, Kao-Walter, Sharon(2019). Numerical Analysis of Pressure Distribution in a Brush Seal based on a 2-D Staggered Tube Banks Model. *Intelligent Automation And Soft Computing* ,25(2):405-411

Mo J, La R J, Anantharam V, (1999). "Analysis and comparison of TCP Reno and Vegas". *Infocom 99 Eighteenth Joint Conference of the IEEE Computer & Communications Societies IEEE, New York, USA*, 4(3):21-25.

Pan Cheng-sheng, Xuan Jing-peng, Wei De-bin, (2012). "Improvement on TCP Vegas Algorithm Based on Base RTT Computation in Satellite Network". *Journal of System Simulation*, 25(6): 1254-1258. (in Chinese)

Wang Jian-feng, Huang Guo-ce, Chen Cai-qiang, (2008). "Enhanced TCP westwood algorithm based on loss differentiation over GEO satellite networks". *Computer Science*, 35(11): 70-73. (in Chinese)

Yue Peng, Zhang Bing, Liu Zeng-Ji, (2006). "A Rerouting Issue with TCP Vegas and its Solution". *Computer Science*, 33(8): 39-41. (in Chinese)

7 NOTES ON CONTRIBUTORS

Qixue Guan, Master of Computer Science, Lecturer, Graduated from Shenyang Ligong University in 2006. Worked at Shenyang Ligong University. Research interests include network transmission technology and intelligent cluster. Network transmission technology shows positive research. Has undertaken several national and provincial projects and several papers have been published

Yueqiu Jiang, Doctor of Computer Science, professor. Graduated from Northeast University in 2004. Worked at Shenyang Ligong University. Research interests include image processing, multimedia applications and satellite communications and signal processing. In the application of multimedia technology shows positive research. Several papers have been published and have undertaken several related projects.