Available online at www.sciencedirect.com
SCIENCE DIRECT
Computer Communications 28 (2005) 429–440
computer communications
www.elsevier.com/locate/comcom

TCP Vegas-A: Improving the Performance of TCP Vegas☆

K.N. Srijith*, Lillykutty Jacob[1], A.L. Ananda

Communication and Internet Research Lab, Department of Computer Science, School of Computing, National University of Singapore, 3 Science Drive 2, Lower Kent Ridge Road, Singapore, Singapore 117543

Received 10 November 2003; revised 19 August 2004; accepted 23 August 2004
Available online 13 September 2004

Abstract

While it has been shown that TCP Vegas provides better performance compared to TCP Reno, studies have identified various issues associated with the protocol. We propose modifications to the congestion avoidance mechanism of the TCP Vegas to overcome these limitations. Unlike the solutions proposed in the past, our solution, named TCP Vegas-A, is neither dependent on optimising any critical parameter values nor on the buffer management scheme implemented at the routers and hence can be implemented solely at the end host. Our simulation experiments over wired as well as over geosynchronous and lower earth orbit satellite links show that TCP Vegas-A is able to overcome several of the identified problems-it can obtain a fairer share of the network bandwidth in wired and satellite scenarios, tackle rerouting issues, rectify Vegas's bias against higher bandwidth flows and prevail over fluctuating RTT conditions of a lower earth orbit satellite link. At the same time, Vegas-A is able to preserve the unique properties of Vegas that had made it a noteworthy protocol.

Keywords: Protocol design; TCP vegas; Congestion avoidance; Satellite links

1. Introduction

Among the several new features implemented in TCP Vegas that was originally proposed by Brakmo et al. [1], one of the most important differences between TCP Vegas and TCP Reno [2] is its congestion avoidance scheme. While TCP Reno (and its variants like NewReno [3]) rely on packet loss detection to detect network congestions, TCP Vegas uses a sophisticated bandwidth estimation scheme to proactively gauge network congestion. TCP Vegas varies it congestion window (cwnd) using the following algorithm

$$\text{cwnd} = \begin{cases} \text{cwnd} + 1 & \text{if diff} < \alpha \\ \text{cwnd} - 1 & \text{if diff} > \beta \\ \text{cwnd} & \text{otherwise} \end{cases}$$

where

- diff = expected_rate − actual_rate
- expected_rate = cwnd(t)/base_rtt, where cwnd(t) is the current congestion window size and base_rtt is the minimum RTT of that connection
- actual_rate = cwnd(t)/rtt, where rtt is the actual round-trip time
- α and β are parameters whose values are usually set as 1 and 3, respectively

The reasoning is that when the actual throughput of a connection approaches the value of the expected maximum throughput, it may not be utilising the intermediate routers' buffer space efficiently and hence should increase the flow rate. On the other hand, when the actual throughput is much less than the expected throughput, the network is likely to be congested and hence the connection should reduce the flow rate.

☆A smaller version of this paper was presented at the IEEE IPCCC 2003 Conference, Phoenix, Arizona.

* Corresponding author. Present address: Department of Computer Science, Faculty of Sciences, Vrije Universiteit, De Boolelaan 1081 A, Amsterdam, The Netherlands. Tel.: +31 611266635; fax: +31 204447653.

E-mail addresses: srijith@alum.comp.nus.edu.sg (K.N. Srijith), lilly@nitc.ac.in (L. Jacob), ananda@comp.nus.edu.sg (A.L. Ananda).

[1] Present Address: Department of Electronic Engineering, National Institute of Technology, Calicut 673601, India.

0140-3664/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2004.08.016

Several studies have established that TCP Vegas does achieve higher efficiency than Reno, causes much fewer packet retransmissions, and is not biased against the connections with longer round trip times (RTTs) [4–6]. However, several problems have also been identified with the protocol. Studies have shown that TCP Vegas performs badly when it competes against TCP Reno for bandwidth [4], is unfair towards older connections [7], does not handle rerouting well and has fairness bias against connections with higher bandwidth [8].

This paper describes our proposed modification to the congestion avoidance mechanism of TCP Vegas, called TCP Vegas-A, to address the issues identified with TCP Vegas. By decomposing TCP Vegas into its individual algorithms and addressing the effect of each of these algorithms on performance, Hengartner et al. [7] had shown that the congestion avoidance mechanism of Vegas has only a minor influence on throughput while at the same time being responsible for the issues identified with the protocol. This observation is one of the motivations to concentrate on the congestion avoidance mechanism to improve the performance of TCP Vegas. Unlike the solutions proposed in the past, TCP Vegas-A is neither dependent on optimising any critical parameter values nor on the buffer management scheme implemented at the intermediate routers and hence can be implemented solely at the end host.

In Section 2, we examine the problems that have been identified with TCP Vegas in detail. In Section 3 we present our modifications to TCP Vegas and the rationale behind the changes is explained. Section 4 presents the experimental results of simulations carried out over a wired network to prove the effectiveness of Vegas-A. In Section 5 we report the results obtained for simulations carried out over satellite links. We conclude in Section 6 by summing up the major findings.

2. Issues with TCP Vegas

In this section, we discuss the issues of TCP Vegas that we have considered in proposing TCP Vegas-A.

2.1. Fairness

TCP Vegas uses a conservative algorithm to decide how and when to vary its congestion window. TCP Reno, in an effort to fully utilise the bandwidth, continues to increase the window size until a packet loss is detected. Thus, when TCP Vegas and TCP Reno connections share a bottleneck link, Reno uses up most of the link and router buffer space. TCP Vegas, interpreting this as a sign of congestion, decreases the congestion window, which leads to an unfair sharing of available bandwidth in favour of TCP Reno [5–7]. This unfairness worsens when router buffer sizes are increased.

Hasegawa et al. [8] proposed TCP Vegas$^+$ as a method to tackle TCP Vegas's fairness issue. However, Vegas$^+$ assumes that an increase in the RTT value is always due to the presence of competing traffic and rules out other possibilities like rerouting. We feel that this is not a reasonable assumption. Furthermore, performance of Vegas$^+$ depends on the choice of optimal value for the new parameter $\text{Count}_{\max}$ introduced in the protocol, which is an open question.

Hasegawa et al. [8] and Raghavendra and Kinicki [9] showed that by using RED routers in place of the tail-drop routers, the fairness between Vegas and Reno can be improved to some degree. But there exists an inevitable trade-off between fairness and throughput, i.e. if the packet dropping probability of RED is set to a large value, the throughput share of Vegas can be improved, but the total throughput is reduced. In [10,11] Feng, Vanichpun and Weigle showed that choosing values of α and β as a function of the buffer capacity of the bottleneck router could improve the fairness condition. However, they do not propose any mechanism to measure this buffer capacity and to set appropriate values for α and β.

2.2. Rerouting

In TCP Vegas, the parameter baseRTT denotes the smallest round-trip delay the connection has encountered and is used to measure the expected throughput. When rerouting occurs in between a connection, the RTT of a connection can change. When the new route has a longer RTT, the Vegas connection is not able to deduce whether the longer RTTs experienced are caused by congestion or route change. Without this knowledge, TCP Vegas assumes that the increase in RTT is due to congestion along the network path and hence decreases the congestion window size [12].

This is exactly opposite of what the connection should be doing. When the propagation delay increases, the bandwidth–delay product (bw*d) increases. The expression (cwnd − bw*d) gives the number of packets in the buffers of the routers. Since the aim of TCP Vegas is to keep the number of packets in the router buffer between α and β, it should increase the congestion window to keep the same number of packets in the buffer when the propagations delay increases.

In [12] the authors also proposed a modification to the Vegas to counteract the rerouting problem by assuming any lasting increase in RTT as a sign of rerouting. Besides the fact that this may not be a valid assumption in all cases, several new parameters K, N, L, δ and γ were introduced in this scheme and finding appropriate values for these variables remain an unaddressed problem.

2.3. Unfair treatment of older connections

As pointed out in [7], TCP Vegas's congestion control mechanism is inherently unfair to older connections. Because of the way baseRTT is calculated, the window

size that would trigger a reduction in congestion window in a TCP Vegas connection is smaller for the older connection than for the newer connection. Similarly, as the critical value of congestion window that triggers an increase in congestion window is smaller for the newer connection, it is more likely to be able to increase its congestion window than the older connection. Hence, the newer connection can achieve higher bandwidth than the older connection.

A closely related problem is the *persistent congestion* problem, where the connections may overestimate the propagation delays and possibly drive the network to a persistently congested state. La et al. [12] studied this problem and suggested that the same solution as the one they proposed for rerouting problem, together with RED routers could solve the persistent congestion problem. Low et al. [13] proposed augmenting Vegas with appropriate active queue management such as Random Exponential Marking for avoiding the persistent congestion problem. However, these solutions rely on queue management mechanism at the intermediate routers, which prevents TCP Vegas from being deployed solely as a user end modification.

2.4. Other issues

Choe and Low showed in [14] that TCP Vegas can become unstable in the presence of network delays and proposes modifications to stabilise the system. In [15] Vanichpun and Feng showed that inappropriate value of γ (a parameter that controls how long TCP Vegas stays in slowstart mode) can lead to slower transient response in TCP Vegas. In [16] Fu and Liew and Vanichpun studied the issues associated with asymmetric networks on the performance of TCP Vegas. We do not address these issues in our paper.

3. TCP Vegas-A, our solution

In this section we present our modification to TCP Vegas. We refer to the modified algorithm as TCP Vegas-A, where 'A' stands for adaptive.

TCP Vegas uses fixed values for variables α and β, set to 1 and 3 usually. Recall that Vegas's strategy is to adjust the source's congestion window in an attempt to keep a small number of packets buffered in the routers along the path. Still subscribing to the idea that the average number of packets in the router buffer is to be kept within α and β, the main idea behind TCP Vegas-A is that rather than fixing α and β, they be made dynamically changeable, adaptive. At the start of a connection, α is set to 1 and β to 3. These values are then changed dynamically depending on the network conditions. Another way of looking at this modification is that we are trying to bring the network probing capability of TCP Reno into TCP Vegas.

While slow start and congestion recovery algorithms of Vegas-A are the same as that of Vegas, we propose a modified congestion avoidance mechanism:

```
01          if β > diff > α {
02                  if Th(t) > Th(t-rtt)  {
03                          cwnd = cwnd +1
04                          α= α+1, β= β+1
05                  }
06                  else if Th(t) <= Th(t-rtt) {
07                          no update of cwnd, α, β
08                  }
09          }
10          else if diff < α {
11                  if α >1 and Th(t) > Th(t-rtt) {
12                          cwnd = cwnd +1
13                  }
14                  else if α >1 and Th(t) < Th(t-rtt) {
15                          cwnd = cwnd –1, α= α-1, β= β-1
16                  }
17                  else if α ==1
18                          cwnd  = cwnd+1
19          }
20                  else if diff > β {
21                          if (α >1) { α= α-1, β= β-1 }
22                          cwnd= cwnd-1
23                  }
24                  else {
25                          no update of cwnd, α, β
26                  }
```

where $Th_{(t)}$ is the actual throughput rate at time t and $Th_{(t-rtt)}$ is the actual throughput rate measured one RTT before t. Note that since TCP Vegas in its present form performs the calculation of the actual throughput rate, to obtain $Th_{(t-rtt)}$ all that is required is an extra internal variable to keep track of the value calculated during previous RTT cycle.

Lines 01–09 kicks in when diff falls between α and β. In the case of TCP Vegas, cwnd is left untouched. However, in Vegas-A rather than be passive, we try to probe the network for available bandwidth.

In line 3 even though diff $> \alpha$, the throughput has been increasing. This is an indication that the network is not fully saturated and that network bandwidth is still available. Hence to probe the network, the sending rate is increased.

In line 4, since the throughput has been increasing over time, diff is decreasing. The existing small values of α and β are preventing the connection from making use of the available bandwidth. Hence, α and β are increased to help congestion window grow. In the present implementation of TCP Vegas-A, α and β are increased and decreased together to maintain their relationship with each other, as in the original implementation.

Lines 10–19 tackle the situation when diff falls below α. In TCP Vegas, cwnd is increased. However, for Vegas-A because of the steps involved in the lines 01–09, we need to perform additional checks before cwnd can be increased.

In Vegas-A, a small value of diff needs not necessarily imply that the bandwidth utilisation is poor. It might be that the dynamically changing value of α has grown to a large value because of which when congestion occurs, even a small throughput can still make diff numerically less than α. Hence at line 15 the cwnd and inflated α and β have to be decreased.

If Vegas-A's diff is greater than β, it could be because of our wrong estimate of α and β. So if $\alpha > 1$, we decrease α and β in lines 20–22.

4. Simulation over wired network

To validate the efficiency of TCP Vegas-A, we simulated various congestion scenarios on wired and satellite networks. The scenarios were selected so as to simulate the conditions that had earlier been identified to cause problems for TCP Vegas. Note that in our simulations, we use TCP NewReno, an enhanced variant of TCP Reno. Simulations with Reno would have resulted in still better performance for Vegas and Vegas-A. Network Simulator 2 (NS2) [17] was used to perform all the simulations in this paper. The TCP/Vegas agent of NS2, which is based on the USC's NetBSD Vegas implementation, was modified to implement Vegas-A.

In this section, we present the results of experiments performed in a wired network. The general topography used in the following experiments is given in Fig. 1.

The data rates and propagation delays of the various links—source-to-router R1, R1-to-R2 (the bottleneck link), and R2-to-destination—are specified in the context of each experiment. Routers use tail-drop policy with a buffer size of 50 packets, unless stated otherwise.

Note that since none of the simulations in this section involves random events (like random packet drops, error, etc.), because of the way NS2 works, repetition of the simulation does not produce varying results. In experiments where multiple flows start off at different times into the simulation, it was seen that changing the arrival times produces similar results and hence we only present one of these cases.

4.1. Rerouting

In this scenario, one TCP Vegas connection is established between S1 and D1. We simulate a change in the route by changing the RTT of the link connecting S1 to R1. The link S1–R1 has a bandwidth of 1 Mbps and initial RTT of 20 ms. After 20 s of file transfer over the connection, the RTT of the link is changed to 200 ms. The links R1–R2 and R2–D1 have bandwidth of 1 Mbps each and RTT of 10 ms for the entire duration of the connection. The simulation was run for 200 s so as to give the connection time to stabilise. Table 1 compares the average throughputs (in bits/s) obtained in the two cases.

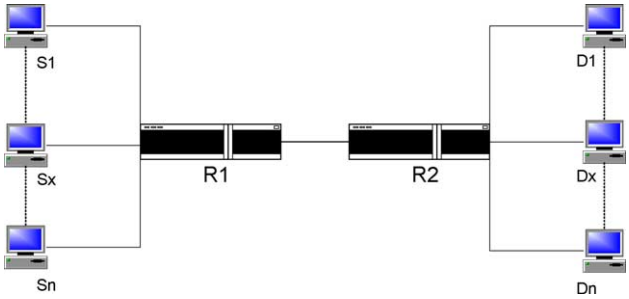

Fig. 1. Network topology used in wired simulations.

Table 1
Throughputs for rerouting simulation

	TCP Vegas	TCP Vegas-A	Difference	%Increase
Throughput	217,320	940,240	722,920	332.7%

Figs. 2 and 3 show the variation of throughputs over the full run. The dotted curve shows the instantaneous throughput, averaged over 0.1 s intervals, and the solid one shows the average throughput. As soon as the RTT changes at 20 s, the instantaneous throughputs of Vegas and Vegas-A start to decrease. However, in the case of Vegas (Fig. 2) the throughput fell all the way down to 0.12 Mbps.

As seen in Fig. 3, the throughput of Vegas-A does suffer initially when the RTT changes at 20 s. As the 'actual throughput' decreases, diff grows large, and hence cwnd is decreased. But when cwnd decreases, expected throughput also decreases. Finally cwnd decreases so much that diff falls between α and β. After that, when there is a slight improvement in the throughput for any particular RTT cycle, the Vegas-A connection increases the value of cwnd, α and β to probe the network. This increase in cwnd in turn increases the throughput further, which in turn triggers an increase in cwnd, α and β again. This process continues until diff becomes less than α. After that, only cwnd increases. This growth of the congestion window is shown in Fig. 4.

The end result is that the available bandwidth is fully utilised and the throughput goes back to the same large

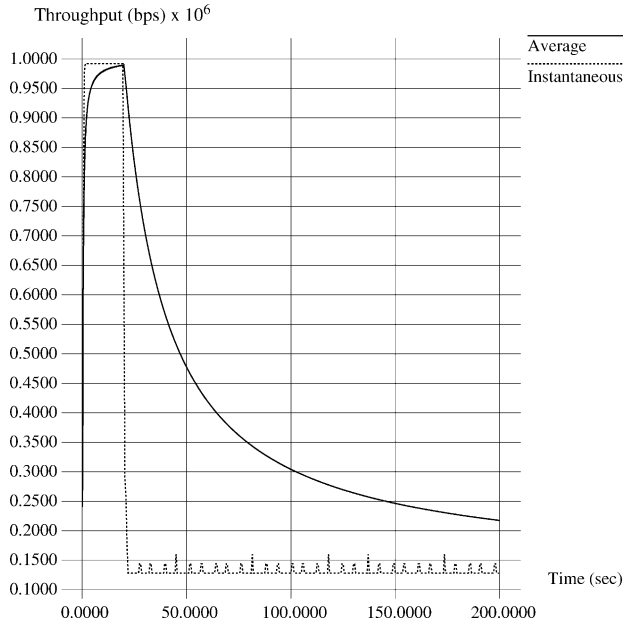

Fig. 2. Throughput variation of Vegas connections due to RTT change.

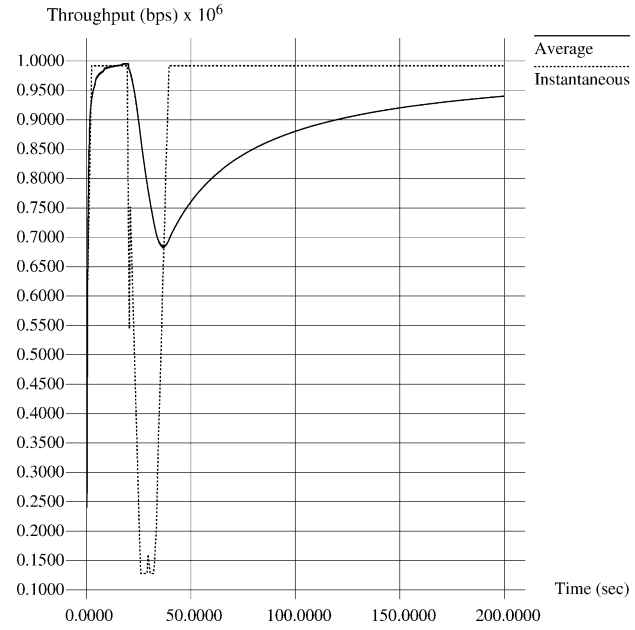

Fig. 3. Throughput variation of Vegas-A connections due to RTT change.

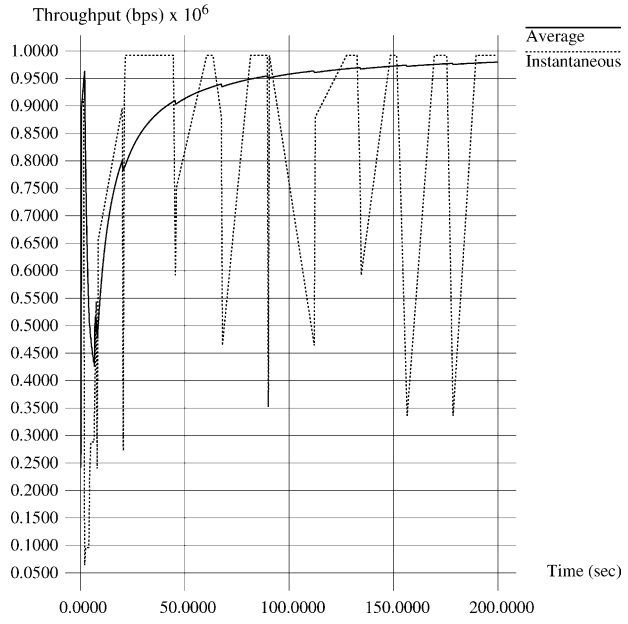

Fig. 5. Throughput variation of NewReno connections due to RTT change.

value prior to the rerouting. Fig. 5 shows the throughput variation for TCP NewReno under similar conditions.

This performance gain of TCP Vegas-A comes at a small 'cost'. The average buffer occupancy at the bottleneck router is larger for Vegas-A than Vegas. Hence the onset of a sudden congestion in the network could cause more packet drop for Vegas-A than for Vegas.

4.2. Bandwidth sharing with NewReno

Next, the performance of Vegas-A when it shares a link with TCP NewReno is considered. There are two connections, S1–D1 and S2–D2. The former uses TCP Vegas-A or Vegas and the latter uses TCP NewReno. The links S1–R1 and S2–R1 are both 8 Mbps with RTT of 20 ms. The bottleneck link R1–R2 is of bandwidth 800 kbps and RTT of 80 ms. R2–D1 and R2–D2 links are both 8 Mbps and RTT 20 ms. S1 was started first and then S2's NewReno traffic was introduced after 10 s.

Fig. 6 shows the instantaneous and average throughputs of the TCP NewReno and TCP Vegas connections as they are competing with each other for a share of the bandwidth. As can be seen from the figure, when TCP NewReno starts at 10 s, it starts to consume Vegas's share of bandwidth.

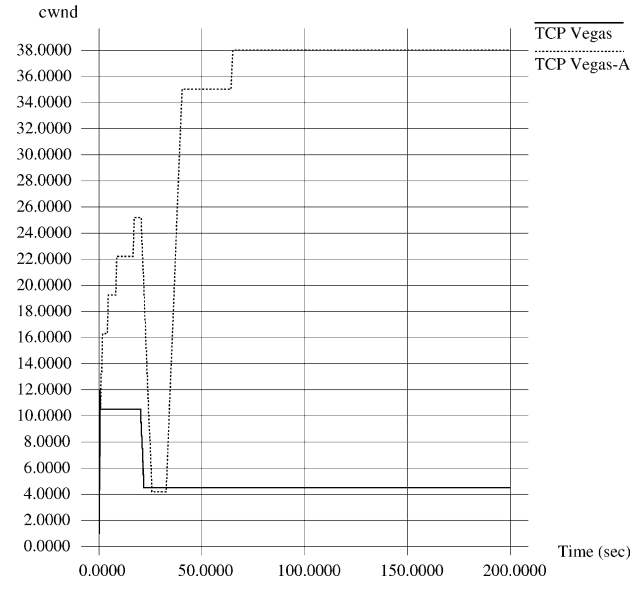

Fig. 4. cwnd variation for Vegas and Vegas-A connections due to RTT change.

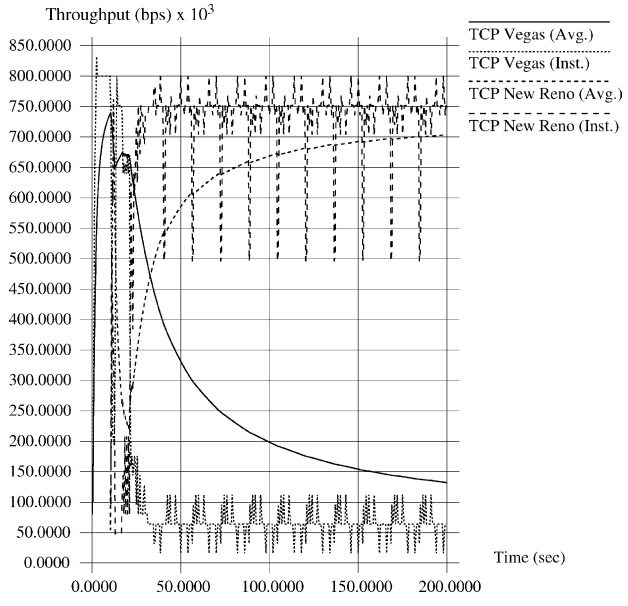

Fig. 6. Throughput of TCP NewReno and Vegas connections over congested link.

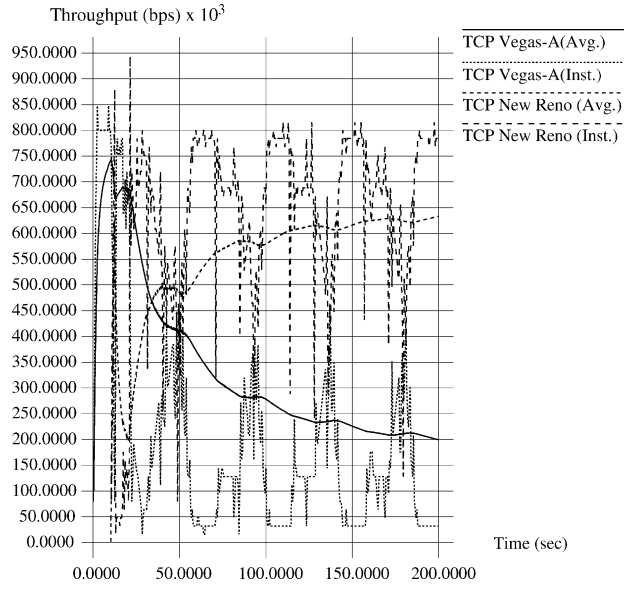

Fig. 7. Throughputs of TCP NewReno and Vegas-A connections over congested link.

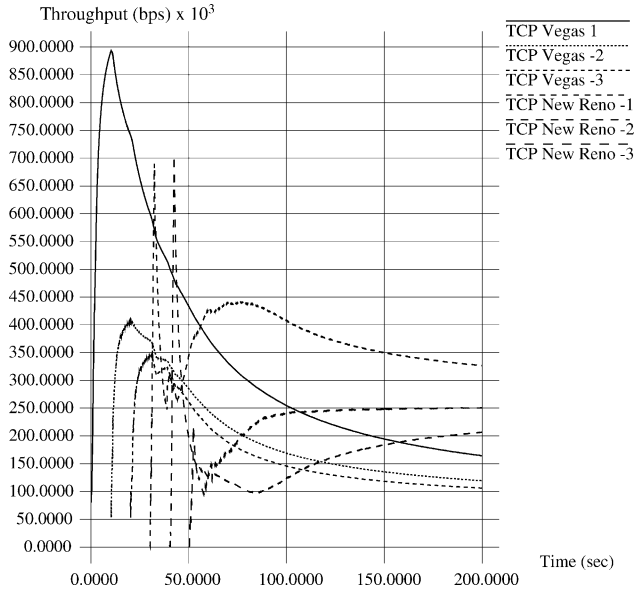

Fig. 8. Throughput of three TCP NewReno and three Vegas connections over congested link.

However, when TCP Vegas is replaced with Vegas-A, the results are very different, as seen in Fig. 7. Even though TCP NewReno gets a more than fair share of the bandwidth, the unfairness is not as much as in the case of TCP Vegas. Table 2 summarises the throughput obtained for each connection. As we can see, the use of Vegas-A reduced the 'unfairness' from a ratio of 5.3:1 to 3.2:1.

This can be explained as follows. When NewReno starts to loose packets, it cuts down its cwnd and thus bandwidth becomes available for taking. Vegas, not being adaptive, does not make use of this opportunity and is stuck at the '$\alpha < \text{diff} < \beta$' state. However, even in that state Vegas-A probes for available bandwidth and increases cwnd at the first available RTT trip. This leads to Vegas-A getting more than fair share of the bandwidth.

When the above experiment was repeated with three NewReno sources and three Vegas/Vegas-A sources, TCP Vegas-A was found getting a more than fair share of the congested link bandwidth. Each source-to-R1 link was 1 Mbps and RTT of 20 ms. R1–R2 link was 1 Mbps and RTT of 80 ms. R2-to-destination link was 1 Mbps and had RTT of 20 ms.

The Vegas/Vegas-A sources (S1, S2, S3) were started at 0, 10 and 20 s while the NewReno sources (S4, S5, S6) were started at 30, 40 and 50 s. Figs. 8 and 9 show the average throughput variation. The average throughput obtained by each flow is given in Table 3.

Table 2
Throughput ratios for NewReno–Vegas and NewReno–Vegas-A connections

NewReno/Vegas	NewReno/Vegas-A
703,875/132,040 = 5.33	633,307/199,320 = 3.17

From the values in Table 3, the *Interprotocol fairness ratio* [18] can be calculated. *Interprotocol fairness ratio* is the ratio of the average bandwidth shares between the two protocols, i.e. it is the ratio of average TCP Vegas/Vegas-A bandwidth calculated across all the Vegas/Vegas-A flows to the average TCP NewReno bandwidth calculated across all the NewReno flows. The *Interprotocol fairness ratio* of the Vegas and NewReno connections can be calculated as 0.497 and that of the Vegas-A and NewReno connections as 2.028. These values show that Vegas-A actually outperforms NewReno.

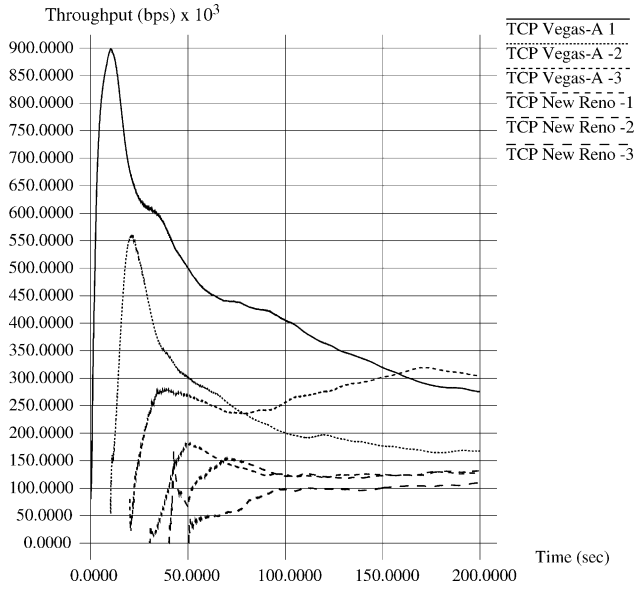

Fig. 9. Throughput of three TCP NewReno and three Vegas-A connections over congested link.

Table 3
Throughputs of NewReno/Vegas and NewReno/Vegas-A connections

Sources	Vegas and NewReno	Vegas-A and NewReno
S1	164,480	275,480
S2	119,242	167,789
S3	106,044	304,178
S4	326,590	127,908
S5	207,002	131,402
S6	250,828	109,336

4.3. Fairness between old and new connections

As mentioned in Section 2.3, TCP Vegas has the disadvantage that newer connection of Vegas enjoys a larger throughput because of the discrepancy in the estimation of RTT of the uncongested link. Vegas-A's modified congestion avoidance mechanism overcomes this shortcoming. To prove this, we simulated a scenario where five TCP Vegas/Vegas-A connections are sharing a link. The source-to-router links were set to 1 Mbps and RTT of 20 ms, while the router-to-destination links had a bandwidth of 1 Mbps and RTT of 100 ms. The sources were started at intervals of 50 s each. Table 4 shows the average throughputs obtained by the connections for the simulation lasting 900 s. Fig. 10 shows the average throughputs of the Vegas connections, while Fig. 11 shows that of the Vegas-A connections.

It is obvious from Fig. 10 that new connections enjoy larger throughputs compared to old connections in the case of Vegas. With Vegas-A, this problem is reduced, as seen in Fig. 11. Table 4 shows that the standard deviation of the average throughput of Vegas-A connections is less than that of Vegas connections. This means that Vegas-A connections have lesser deviation from mean value and thus share the bandwidth in a fairer manner among each other compared to the Vegas connections.

4.4. Bias against high bandwidth flows

Hasegawa et al. [8] showed that TCP Vegas, like Reno, has the fairness bias against connections with higher bandwidth. To test whether Vegas-A can perform better than Vegas, simulations were conducted with three connections S1–D1, S2–D2, and S3–D3, and with the S1–R1 link bandwidth limited to 128 kbps, S2–R1 to 256 kbps, and S3–R1 to 512 kbps. The RTT of each source-to-R1 link

Table 4
Throughputs of five Vegas and Vegas-A connections

Sources	Vegas	Vegas-A
S1	218,531	221,447
S2	191,533	199,760
S3	206,176	247,431
S4	247,585	229,577
S5	266,913	234,662
Std. deviation	33,217.1	17,711.1

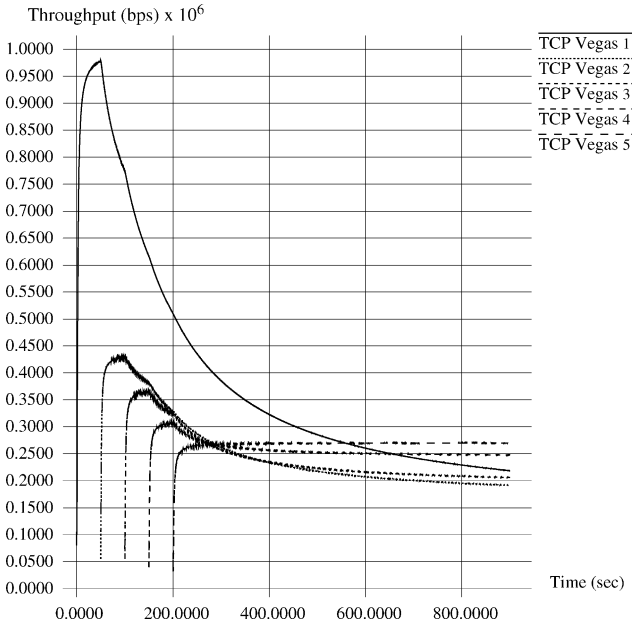

Fig. 10. Throughput of five Vegas connections over congested link.

was set to 5 ms. R1–R2 link was bandwidth limited to 400 kbps and the RTT was set to 10 ms. R2-to-destination link had a bandwidth of 1 Mbps and RTT of 5 ms. The simulations were carried out for a period of 1800 s. Table 5 summarises the throughputs (in kbps) obtained and compares it with the expected values. With the Si–R1 link bandwidth represented by bw_i, the expected value for connection Si–Di is calculated as $bw_i * R / \sum_1^3 bw_i$ where R is the bottleneck link bandwidth.

Ideally, S3–D3 connection should have received 228 kbps of the bottleneck bandwidth, but when Vegas is used, the connection ends up with just 120.5 kbps

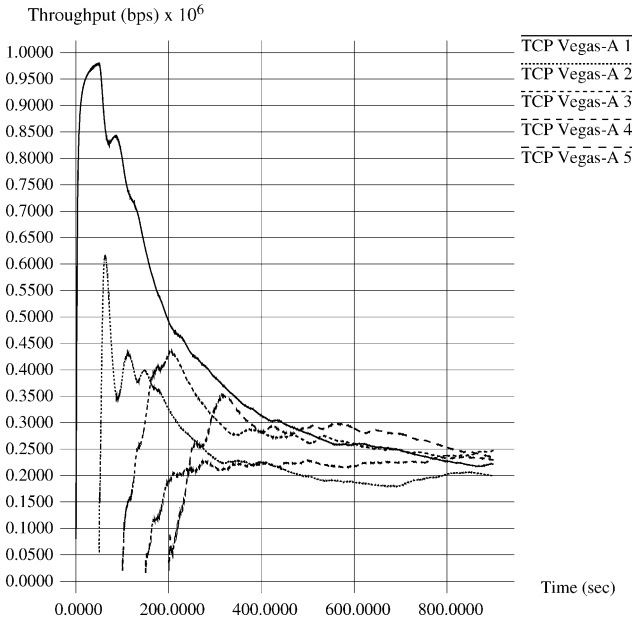

Fig. 11. Throughput of five Vegas-A connections over congested link.

Table 5
Comparison of Vegas and Vegas-A bias against high bandwidth connection for 1800 s

	S1	S2	S3
Expected	57.14	114.29	228.57
Vegas	123.34	146.85	120.46
Vegas-A	98.90	134.54	158.25

Table 6
Comparison of NewReno, Vegas and Vegas-A connection on a 20 ms RTT link

Source	5 MB file			10 MB file		
	Time	Avg.	Retx.	Time	Avg.	Retx.
NewReno	162.353	11.46	58	322.353	23.27	62
Vegas	160.253	0.65	0	320.253	1.3	0
Vegas-A	160.253	5.62	0	320.256	12.4	0

bandwidth. However, when the source is switched to Vegas-A, the three connections enjoy much fairer shares of the bandwidth. This is because, in the case of Vegas, the cwnd attains a steady value, such that $\alpha < \text{diff} < \beta$, in the very early stage of the connection, and remains in that state for the entire duration of the connection. In the case of Vegas-A, α and β can vary dynamically, allowing cwnd to probe for more bandwidth share, resulting in Vegas-A exhibiting fairer properties.

4.5. Retaining properties of Vegas

TCP Vegas-A tries to improve upon the congestion avoidance mechanism of TCP Vegas by trying to adapt to the availability of the network bandwidth and in the process makes it slightly more aggressive than Vegas. However, it is critical that in doing so, Vegas-A should not lose the unique properties of Vegas that lead to lower router buffer utilisation and lesser packet retransmissions.

To ensure that Vegas-A does retain the properties of Vegas, namely fewer packet drops and lower average buffer occupancy at the routers simulations were conducted with only one Vegas/Vegas-A/NewReno source connected through the routers to the destination. The S1–R1 link bandwidth was set to 1 Mbps and the RTT of the link to 5 and 45 ms, respectively, for two sets of experiments. The R1–R2 bandwidth was restricted to 250 kbps and the RTT to 5 and 45 ms, respectively, for the two sets of experiments. The R2–D2 link bandwidth was set to 1 Mbps and RTT to 10 ms. File sizes of 10 and 5 MB were sent from source to the destination. Table 6 shows the values recorded for the time taken for the complete transfer, the average queue length at the router (in packets), and the number of retransmitted packets.

As the table show, Vegas and Vegas-A outperform NewReno in all performance measures, as expected. The only difference between Vegas and Vegas-A is that the average buffer occupancy at the routers is larger for Vegas-A compared to Vegas. This is expected since Vegas-A adapts α and β to values larger than 1 and 3, which in turn let Vegas-A increase the congestion window to a larger value and thus pushing in more data into the network. However, note that this larger average buffer occupancy which is still much lower than buffer size at the router does not increase the time required to complete the transfer of the files, thus showing that the queuing delay is not increased much.

Next we studied the performance of Vegas, Vegas-A and NewReno when the router queue sizes were varied. The RTT of the source to destination links were fixed at 40 ms. S1–R1 and R2–D1 link bandwidths were set at 1 Mbps, while the R1–R2 link bandwidth was set at 500 kbps. Files of size 10 MB was sent from S1 to D1. Table 7 shows the values obtained. The unit of time is seconds and 'Retx.' records the number of packets retransmitted.

The results show that when the buffer size is as small as 10, 15 or 20, Vegas-A retransmits at the most 1 or 2 packets, while Vegas does not loose any packets at all. This number is extremely small compared to the number of packets dropped and retransmitted when NewReno is used. Furthermore, when the queue size is increased to 25 and 30, the behaviour of Vegas-A approaches that of Vegas. We feel that this very small packet drop of Vegas-A is acceptable given the performance increase Vegas-A provides.

5. Satellite links

Several studies have been carried out to investigate the performance of TCP over satellite links. As these satellite systems are being envisaged to provide affordable 'last-mile' network access to home and small business users worldwide, it is important to understand how the properties

Table 7
Comparison of NewReno, Vegas and Vegas-A connections with different router buffer queue size

Source	Buffer size									
	10		15		20		25		30	
	Time	Retx.	Time	Retx.	Time	Retx.	Time	Retx.	Time	Retx.
NewReno	161.3	106	161.6	74	161.9	61	162.2	56	162.5	55
Vegas	160.4	0	160.4	0	160.4	0	160.4	0	160.4	0
Vegas-A	160.5	2	160.6	1	160.6	1	160.4	0	160.4	0

of these links affect the transport level protocols and the efficiency of these protocols over such links. In particular, two types of advanced satellite systems are being developed: high power satellite deployed at geostationary orbits (GEO), and large constellations of satellites deployed at lower earth orbits (LEO).

In this section, we mention some unique characteristics of satellite links and report the results of the simulations conducted over GEO and LEO links. To our knowledge, this is the first paper that has studied the performance of TCP Vegas over satellite links. In the following sections, we evaluate the performance of TCP Vegas and TCP Vegas-A over LEO and GEO satellite links. The links were simulated using the extensions provided in NS2.

5.1. Satellite link characteristics

5.1.1. Latency

TCP connections over satellite links experience RTT many orders more than that experienced on wired networks. For connections traversing GEO links, the RTT can be as large as 520 ms, and may be more if interleavers are inserted in the path to perform forward error correction. However, fluctuations in these RTTs are not usual. In the case of LEO links, due to its lower altitudes, the RTTs experienced are smaller and can be as low as 40 ms for each satellite used in the connection. However, RTT variations are fairly common in LEO systems due to the relative motion of the satellites.

5.1.2. Transmission errors

Satellite links have been reported to experience varying bit error rates (BER) due to the use of legacy equipments; as low as 10^{-7} on average and 10^{-4} worst case [19]. TCP has the inherent limitation of not being able to distinguish between packets lost due to congestion and that due to corruption, classifying all losses as being due to congestion. Thus, a high BER causes TCP connections to incorrectly reduce its congestion window (cwnd) and in the process, attaining lower throughput.

5.1.3. Large bandwidth-delay product

Due to the large RTT experienced by TCP connections on satellite links, the bandwidth delay product (bw*d) of these connections tend to be large. To fully utilise the network, a TCP connection's cwnd should be almost equal to the bw*d of the link. Even though RFC 1323 [20] has defined windows scaling options that allow the use of large cwnd, they are rarely used practically because, to trigger the use of window scaling options, the connection must request large sending and receiving buffer sizes. Most TCP implementations use small buffer sizes, even as low as 4 KB [21]. Thus, a TCP connection that does not use large send and receive buffer sizes and that which is not able to increase its cwnd to a large value will experience smaller throughputs in a satellite network.

To get a complete picture of the performance, various scenarios were simulated with varying number of connections and packet error rates (PER).

5.2. GEO satellite link

In this set of simulations, Vegas-A's performance over a GEO satellite link between New York and San Francisco was analysed. The uplink and downlink bandwidth was set to 1.5 Mbps. The terminals were set to use drop tail queue of 50 packets buffer size. FTP traffic was generated using NewReno, Vegas and Vegas-A, from New York to San Francisco. Scenarios with varying PER were simulated with single and competing traffics. For each value of PER, the simulation was performed 10 times using different random seed values for error generation and the average of the results obtained were noted. Variation in result for these simulation runs were found to be too small to be noted.

5.2.1. One connection, varying PER

One FTP source was run while the PER of the satellite links were varied. The simulation was run for 600 s to give the connection time to tide over transient conditions. Table 11 shows the results obtained when various TCP versions were used as the FTP source. The throughputs are in bits/s and number of retransmission in units of packets.

As can be seen from Table 8, Vegas and Vegas-A perform almost identically and their throughputs are higher than that of TCP NewReno's. Note that the higher amount of packet drops in case of PER >0 for Vegas and Vegas-A is because when throughput is higher, more packets are sent into the network and hence more packets are corrupted. The packets lost by Vegas and Vegas-A is not due to congestion, as the values obtained for PER of

Table 8
Throughput and number of retransmissions for NewReno, Vegas and Vegas-A connections using GEO satellite links

PER	NewReno		Vegas		Vegas-A	
	Throughput	Retx.	Throughput	Retx.	Throughput	Retx.
0	1,208,387	163	1,444,300	0	1,444,474	0
0.0005	675,312	189	1,097,909	53	1,098,457	52
0.005	263,907	112	407,709	149	407,759	149
0.05	64,828	239	103,131	371	102,557	371

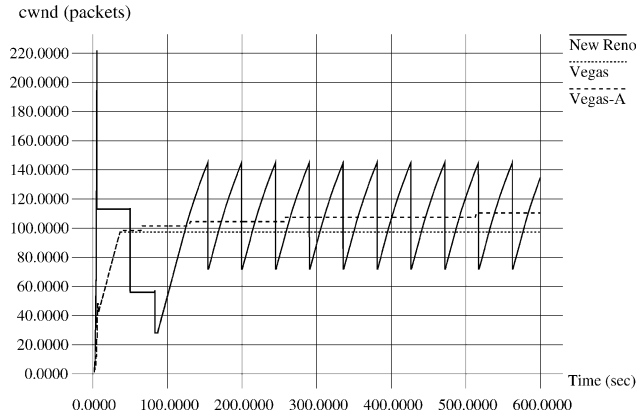

Fig. 12. cwnd variation for NewReno, Vegas and Vegas-A connections with PER = 0.0.

0.0 simulation shows. The better segment loss detection algorithm of Vegas helps Vegas and Vegas-A connections perform lesser number of slow starts and thus help in achieving better throughput. This will be further investigated in the following sections.

Fig. 12 shows the cwnd variation for PER = 0.0. The cwnd of NewReno shows the typical saw tooth format, while Vegas's cwnd remains constant at around 100 packets. However, just like in the case of the wired network, Vegas-A's cwnd keeps on increasing slowly but steadily without any fall. As a result, Vegas and Vegas-A perform better than NewReno in terms of throughput and number of retransmissions.

5.2.2. Competing connections, varying PER

The effect of presence of competing NewReno traffic on the performance of the TCP connections on GEO links are discussed later. A NewReno/Vegas/Vegas-A connection was started in the beginning and after 10 s, another NewReno traffic was introduced. FTP applications were attached to both the sources. The simulations were run for 600 s to make sure that the results obtained were at stable conditions. Random seeds were used for packet corruption probabilities and average values of several runs were used for the recording of values in Tables 9 and 10. It details the various throughputs obtained for various TCP source combinations and PERs.

In Table 9, when PER is very small, the competing NewReno flow obtained a more than fair share of the bandwidth. However, it is worth noting that Vegas-A flow was able to compete better than Vegas and achieve higher throughput. This result is similar to the result presented for the wired networks and is due to the ability of Vegas-A connections to increase the cwnd to a larger value than what Vegas can.

When PER was increased (Table 10), the competing NewReno flow's throughput decreased to a value lesser than that of the competing Vegas or Vegas-A connection. Again, Vegas-A connection is able to achieve a better throughput compared to Vegas. This can be explained by the fact that the enhanced loss detection mechanism of TCP Vegas (and

Table 9
Throughput and goodput for competing connections at PER 0.0005

		Through-put (bps)	Goodput (bps)	Lost (packets)
NewReno vs NewReno	NewReno	526,491	523,993	187
	NewReno	660,461	658,807	122
Vegas vs NewReno	Vegas	552,440	552,146	22
	NewReno	734,386	732,285	155
Vegas-A vs NewReno	Vegas-A	592,854	592,520	25
	NewReno	724,678	722,997	124

Table 10
Throughput and goodput for competing connections at PER 0.05

		Through-put (bps)	Goodput (bps)	Lost (packets)
NewReno vs NewReno	NewReno	62,224	58,885	250
	NewReno	67,824	64,597	238
Vegas vs NewReno	Vegas	93,515	88,534	373
	NewReno	64,976	61,776	236
Vegas-A vs NewReno	Vegas-A	94,797	89,709	381
	NewReno	64,881	41,410	256

Table 11
Slow starts performed by competing connections at PER 0.05

		Slowstarts
Vegas vs NewReno	Vegas	4
	NewReno	31
Vegas-A vs NewReno	Vegas-A	2
	NewReno	32

Table 12
Throughput of various TCP connections over LEO satellite with 0.0 PER

	Throughput (bps)	Lost packets
Vegas	1,348,086	0
Vegas-A	1,459,432	52
NewReno	1,455,693	189

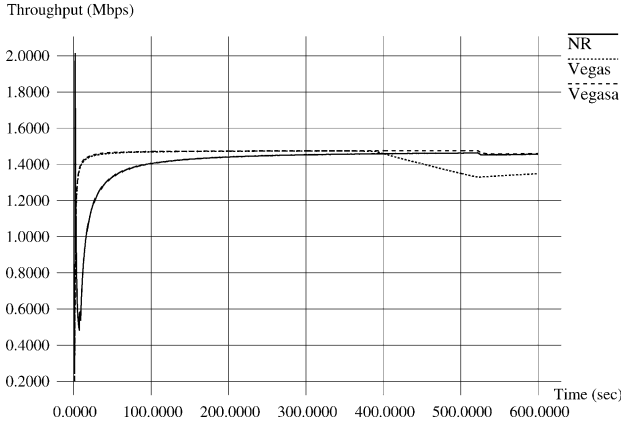

Fig. 13. Average throughput of NewReno, Vegas and Vegas-A connections over LEO links.

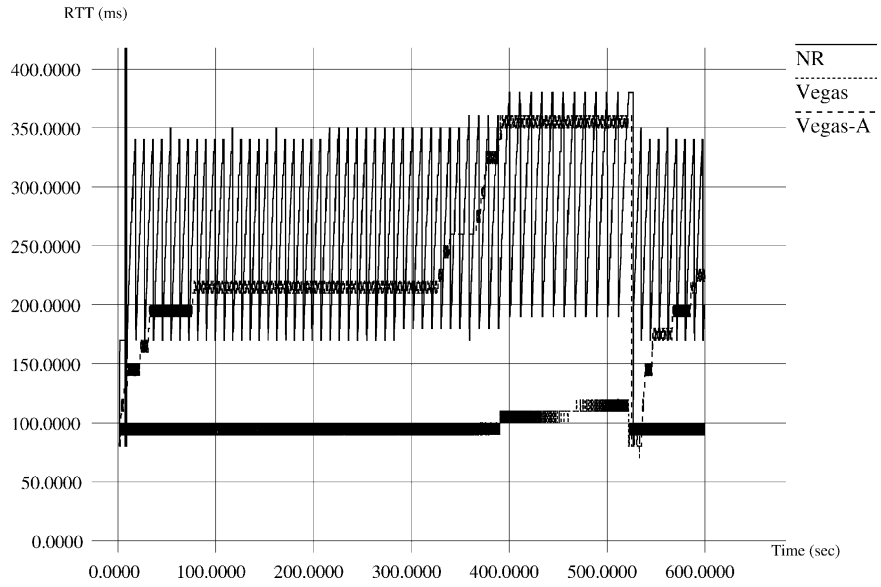

Fig. 14. RTT variation of NewReno, Vegas and Vegas-A connections over LEO links.

Vegas-A) is able to prevent the triggering of slowstarts on several occasions as can be seen in Table 11.

5.3. LEO satellite links

The Iridium satellite system was chosen as an example of a LEO satellite link. The satellites are at an altitude of 780 km above earth, with an orbital period of 6206.9 s, inter-satellite separation of 32.72° and seam separation of 22°. The two connection terminals were simulated as being situated at Berkeley and Boston. Various scenarios were simulated and each simulation was run for 600 s.

5.3.1. Single source

A single source of TCP connection was simulated to carry FTP data over the links assuming no errors in the link. NewReno, Vegas and Vegas-A traffic was simulated and throughput attained was recorded in Table 12.

As can be seen from Table 12, Vegas-A's throughput is the largest, while Vegas's is the lowest. Fig. 13 shows that until about 375 s, Vegas's and Vegas-A's performance are comparable. However, from 375 to around 525 s, Vegas's throughput decreases considerably. It begins to pick up slowly again after 525 ms, while the performance of Vegas-A and NewReno remains mostly steady and increases at certain time intervals. This observation can be explained as follows.

The relative motion between satellites in a LEO system causes handoff that fluctuates the RTT of the connection. This fluctuating RTT of a LEO satellite system creates a situation similar to a route change in a wired network, an issue that has been identified as a problem area for the performance of Vegas. At around 375 s into the simulations,

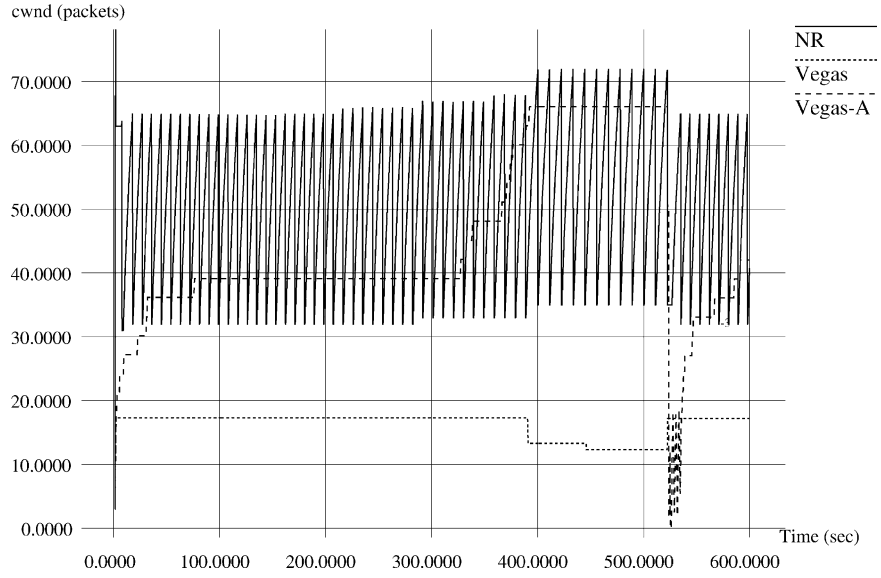

Fig. 15. cwnd variation of NewReno, Vegas and Vegas-A connections over LEO satellite links.

Table 13
Throughput of various TCP connections when competing with TCP NewReno source over LEO satellite

		Through-put (bps)	Goodput (bps)	Lost (packets)	% Share of good-put
NewReno vs New-Reno	NewReno	742,878	740,114	207	47.30
	NewReno	734,509	732,610	140	52.7
Vegas vs NewReno	Vegas	154,484	154,417	50	9.80
	NewReno	1,340,244	1,338,156	154	90.20
Vegas-A vs New-Reno	Vegas-A	387,005	386,337	50	24.2
	NewReno	1,104,719	1,102,712	148	75.8

the RTT of the connections starts to increase as seen in Fig. 14. As in the case of wired network, Vegas assume that this increase in RTT is due to congestion and reduces the cwnd. But Vegas-A, in a manner similar to the wired scenario is able to alter values of α and β and thus increase cwnd during the period of growing RTT. This larger cwnd during the period from 375 to 525 s increases the overall throughput of the Vegas-A connection (Fig. 15).

5.3.2. Competing traffic

Competing traffic scenario was simulated next to see how Vegas and Vegas-A fares when it competes with a NewReno connection for the available bandwidth. The competing NewReno connection starts 10 s after the first connection was established. It was found that changing the order of connections yielded similar results.

As evident from the numbers in Table 13, Vegas performs disastrously when it has to compete with New-Reno connection, getting as low as 10% of the total bandwidth. However, Vegas-A is able to perform better and gets as much as 24% of the bandwidth. The situation is similar to that in wired network when Vegas/Vegas-A competes with NewReno connections, and hence this result is expected and can be explained using the same reasoning.

When the PER was varied in the LEO system, the behaviour observed were similar to that of the GEO system. Hence, they have not been reported here to avoid repetitiveness.

6. Conclusion

In this paper, we looked at the problems associated with TCP Vegas and proposed modifications to the congestion avoidance algorithm of TCP Vegas to overcome these limitations. Our modification, TCP Vegas-A, was shown to perform better than TCP Vegas in both wired and satellite networks. We showed by simulations that TCP Vegas-A is able to compete better against TCP NewReno, overcome rerouting conditions in wired and fluctuating RTT in satellite networks and overcome bias against high bandwidth and older connections, while at the same time retaining the useful properties of TCP Vegas.

References

[1] L.S. Brakmo, S.W. O'Malley, L.L. Peterson, TCP Vegas: new techniques for congestion detection and avoidance, Proceedings of ACM SIGCOMM'94, 1994 pp. 24–35.
[2] V. Jacobson, Congestion avoidance and control, Computer Communications Review 18 (4) (1988) 314–329.
[3] S. Floyd, T. Henderson, The NewReno modification to TCP's fast recovery algorithm, RFC 2582 April 1999;.
[4] L.S. Brakmo, L.L. Peterson, TCP Vegas: end to end congestion avoidance on a global internet, IEEE Journal on Selected Areas in Communications 13 (8) (1995) 1465–1480.
[5] J.S. Ahn, P. Danzig, Z. Liu, L. Yan, Evaluation of TCP Vegas: emulation and experiment, in: Proceedings of ACM SIGCOMM'95, August 1995, pp. 185–195.
[6] J. Mo, R.J. La, V. Anantharam, J. Walrand, Analysis and comparison of TCP Reno and Vegas, Proceedings of IEEE INFOCOMM'99, March 1999 pp. 1556–1563.
[7] U. Hengartner, J. Bolliger, Th. Gross, TCP Vegas Revisited, Proceedings of IEEE INFOCOM'2000, March 2000 pp. 1546–1555.
[8] G. Hasegawa, K. Kurata, M. Murata, Analysis and improvement of fairness between TCP Reno and Vegas for deployment of TCP Vegas to the internet, Proceedings of the IEEE International Conference on Network Protocols (ICNP 2000) November 2000;.
[9] A.M. Raghavendra, R.R. Kinicki, A simulation performance study of TCP Vegas and random early detection, Proceedings of IPCCC'99, February 1999 pp. 169–176.
[10] E. Weigle, W. Feng, A case for TCP Vegas in high-performance computational grids, Proceedings of Ninth International Symposium on High Performance Distributed Computing August 2001;.
[11] W. Feng, S. Vanichpun, Enabling compatibility between TCP Reno and TCP Vegas, IEEE Symposium on Applications and the Internet (SAINT 2003) January 2003;.
[12] R.J. La, J. Walrand, V. Anantharam, Issues in TCP Vegas, July 1998, available at http://www.path.berkeley.edu/~hyongla.
[13] S. Low, L. Peterson, L. Wang, Understanding TCP Vegas: a duality model, Proceedings of ACM SIGMETRICS 2001, June 2001 pp. 226–235.
[14] D.H. Choe, S.H. Low, Stablized vegas, Proceedings of IEEE INFOCOMM 2003 April 2003;.
[15] S. Vanichpun, W. Feng, On the transient behavior of TCP Vegas, 11th IEEE International Conference on Computer Communications and Networks October 2002;.
[16] C.P. Fu, S.C. Liew, A remedy for performance degradation of TCP Vegas in asymmetric networks, IEEE Communications Letters Jan (2003).
[17] Network Simulator 2 (NS2), http://www.isi.edu/nsnam/ns.
[18] R. Rejaie, M. Handley, D. Estrin, RAP: an end-to-end rate-based congestion control mechanism for realtime streams in the Internet, Proceedings of IEEE INFOCOM'99, March 1999 pp. 1337–1345.
[19] T. Henderson, R. Katz, Transport protocols for internet-compatible satellite networks, IEEE Journal on Selected Areas of Communications Feb (1999).
[20] V. Jacobson, R. Braden, D. Borman, TCP extensions for high performance, RFC 1323 May 1992;.
[21] J. Heidemann, Performance interactions between P-HTTP and TCP implementations, ACM Computer Communications Review 27 (2) (1997) 65–73.