Volume 3, Issue 7, July 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

TCP PIPE Algorithm for Obtaining high Throughput by using Dynamic Change of Congestion Window and Effective Control of Congestion in WSN's

Dr.R.Seshadri
Prof & Director of University Computer Center
S.V.University , Tirupati, India

Prof.N..Penchalaiah
Department of Computer Science Engineering
ASCET , Gudur, India

Abstract— ***TCP Vegas may considered as a better alternate congestion control algorithm over TCP Reno, it has still its own issues when it is implemented in wide area emulator. TCP Vegas still has a few obstacles for it to be deployed in the Internet. Studies have shown unfair treatment to Vegas connections when they compete with Reno connections. Other issues identified with TCP Vegas are problems of rerouting, persistent congestion, and discrepancy in flow rate tied with starting times and link bandwidth. Here these issues were investigated and proposed modifications to the congestion avoidance mechanisms of the TCP Vegas, with the slow start and congestion recovery algorithms of Vegas remaining untouched. Our solution is neither dependent on any other critical dependent parameter values nor on the buffer management scheme at the routers (e.g., RED). Our experiments show that the proposed TCP PIPE (modified version of TCP Vegas) is able to obtain a fairer share of the network bandwidth when competing with other TCP flows. At the same time, our experiments prove that TCP PIPE preserves the properties of Vegas that have made it a striking protocol.***

Keywords— TCP Vegas, TCP Reno, TCP PIPE, Congestion Window, round trip times.

I. INTRODUCTION

TCP Vegas, a fundamentally different congestion control scheme from that TCP Reno. Several studies were made on TCP Vegas. These studies have established that Vegas does achieve higher efficiency rather than Reno, causes much fewer packet retransmissions, and is not biased against the connections with longer round trip times (RTTs), but does not receive a fair share of bandwidth when competing with Reno connections[2,3,4]. By decomposing TCP Vegas into its individual algorithms an d addressing the effect of each of these algorithms on performance, Hengartner et el. [5] have shown that the reported performance gain are achieved primarily by TCP Vegas new techniques for slow start and congestion recovery, and that the congestion avoidance mechanism of Vegas has only a minor influence on throughput while also being responsible for the fairness problems. Hasegawa et el. [6] and Raghavendra et el. [7] show that with the RED routers in place of the tail-drop routers, the fairness between Vegas and Reno can be improved to some degree. But there exists 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 degraded. In [8], Richard J.La identifies the problems associated with TCP Vegas' use of estimate of RTT as a decision variable. In particular, they identify the issue of rerouting, the persistent congestion problem, and the discrepancy in flow rates tied with starting times of different Vegas connections. They propose a solution to overcome the problem of rerouting and show that persistent congestion could also be solved using the combination of the same modification and RED gateways. However, finding appropriate threshold values for RED gateways and finding the appropriate parameter values for the proposed modifications of TCP Vegas congestion avoidance mechanism are open problems. TCP PIPE addresses the issues of routing and bias against older connections as well. Further our experiment show that TCP PIPE preserves the proven nice properties of TCP Vegas. In this paper modifications were proposed to the congestion avoidance mechanism of TCP Vegas to address the fairness issues with our entire modified version TCP PIPE. Section 2 briefly introduces the congestion avoidance mechanism of TCP Vegas. The issues related to the present implementation of TCP Vegas are presented in section 3. In section 4, modifications to TCP Vegas are proposed. In section 5 experimental results to show that the proposed modification does perform better under the scenarios outlined in section 3 and discuss these results. And the conclusion is in section 6.

II. INTRODUCTION TO TCP VEGAS

Following a series of congestion collapses starting in October of '86, Jacobson and Karels developed a congestion control mechanism, which was later named TCP Tahoe [5]. Since then, many modifications have been made to TCP, and different versions have been implemented such as TCP Tahoe and Reno. TCP Vegas was first introduced byBrakmo et al. in [2]. There are several changes made in TCP Vegas. First, the congestion avoidance mechanism that TCP Vegas uses is quite different from that of TCP Tahoe or Reno. TCP Reno uses the loss of packets as a signal that there is congestion in the network and has no way of detecting any incipient congestion before packet losses occur. Thus, TCP Reno reacts to congestion rather than attempts to prevent the congestion.

TCP Vegas, on the other hand, uses the difference between the estimated throughput and the measured throughput as a way of estimating the congestion state of the network. The algorithm briefly described here. For more details on TCP Vegas, refer to [2]. First, Vegas sets BaseRTT to the smallest measured round trip time, and the expected throughput is computed according to

$$Expected = \frac{WindowSize}{BaseRTT}$$

Where 'WindowSize' is the current window size. Second, Vegas calculate the current Actual throughput as follows. With each packet being sent, Vegas records the sending time of the packet by checking the system clock and computes the round trip time (RTT) by computing the elapsed time before the ACK comes back. It then computes Actual throughput using this estimated RTT according to

$$Actual = \frac{WindowSize}{RTT}$$

Then, Vegas compares Actual to Expected and computes the difference

Diff=Expected-Actual

The above equation is used to adjust the window size. Note that Diff is non-negative by definition. Define two threshold values, $0 \leq \alpha \leq \beta$. If Diff < α, Vegas increases the window size linearly during the next RTT. If Diff > β, then Vegas decreases the window size linearly during the next RTT. Otherwise, it leaves the window size unchanged. What TCP Vegas attempts to do is as follows. If the actual throughput is much smaller than the expected throughput, then it suggests that it is likely that the network is congested. Thus, the source should reduce the flow rate. On the other hand, if the actual throughput is too close to the expected throughput, then the connection may not be utilizing the available flow rate, and hence should increase the flow rate. Therefore, the goal of TCP Vegas is to keep a certain number of packets or bytes in the queues of the network [2, 9]. The TCP Vegas mechanism is as follows.

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

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 roundtrip time
- α and β are parameters whose values are usually set as 1
- and 3, respectively

Now note that this mechanism used in Vegas to estimate the available bandwidth is fundamentally different from that of Reno, and does not purposely cause any packet loss. Consequently this mechanism removes the oscillatory behavior from Vegas, and Vegas achieve higher average throughput and efficiency. Moreover, since each connection keeps only a few packets in the switch buffers, the average delay and jitter tend to be much smaller.

Another improvement added to Vegas over Reno is the retransmission mechanism. In TCP Reno, a rather coarse grained timer is used to estimate the RTT and the variance, which results in poor estimates. Vegas extend Reno's retransmission mechanism as follows. As mentioned before, Vegas record the system clock each time a packet is sent. When an ACK is received, Vegas calculate the RTT and use this more accurate estimate to decide to retransmit in the following two situations [2]:

i) When it receives a duplicate ACK, Vegas checks to see if the RTT is greater than timeout. If it is, then without waiting for the third duplicate ACK, it immediately retransmits the packet.

ii) When a non-duplicate ACK is received, if it is the first or second ACK after a retransmission, Vegas again checks to see if the RTT is greater than timeout. If it is, then Vegas retransmit the packet.

III. Issues with TCP Vegas

Even though TCP Vegas has been shown to provide better performance in terms of increased throughput, reduced jitter and much reduced retransmissions, there are a number of issues associated with TCP Vegas that prevent it from being accepted and implemented on the internet. In this section some important issues were discussed and also there are so many issues that exist in TCP Vegas those are not addressed in this paper.

The main issues with TCP Vegas are:

(i) Fairness

(ii) Rerouting
(iii)Unfair treatment of old connection

(i). Fairness

TCP Vegas uses a conservative algorithm to decide when to increase/decrease the congestion window. However, TCP Reno, in an effort to fully utilize the bandwidth, continues to increase the window size until a packet loss is detected thus, while TCP Vegas tries to maintain a smaller queue, TCP Reno and its variants keep a larger number of packets in the buffer on the average, and thus steal a larger bandwidth.

When the TCP Vegas and TCP Reno connections shared a bottleneck link, Reno uses up most of the router buffer space. TCP Vegas, interpreting this as a sign of congestion, decreases the congestion window, which leads to an unfair share of bandwidth for TCP Vegas. Furthermore, when the router buffer size is large, the average window size of TCP Reno connections will be large, while the window size of TCE' Vegas connections will remain unchanged, as it does not inflate the window size larger than 8. This means, the larger the router buffer size, the worse the fairness between Reno and Vegas connections.

Hasegawa et el [5]. The proposed TCP Vegas+ as a method to tackle TCP Vegas fairness issue. Vegas' has two modes. In the moderate mode, Vegas' behaves identically to the normal TCP Vegas, but it enters the aggressive mode to increase its window size aggressively (identical to TCP Reno) when it detects the presence of competing Reno traffic. 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. Furthermore, performance of Vegas' depends on the choice of optimal value for $count_{max}$ in order to switch between the two modes.

(ii). Rerouting

In Vegas, the parameter baseRTT denotes the smallest round-trip delay and is used to measure the expected throughput when a route is changed, the RTT of a connection can change. Vegas is not usually affected if the new route has a smaller RTT, as baseRTT will be updated But when the new route has a longer RTT, the connection will not be able to deduce whether the longer RTTs experienced are because of congestion or route change. Without this knowledge, TCP Vegas assumes that the increase in RTT is because of congestion along the network path and hence decreases the congestion window size.

This is exactly opposite of what the connection should be doing: When the propagation delay increases, the bandwidth-delay product (*bw*d*), which is an estimate of the number of packets that can be held in the network, increases. For any connection, 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 between α and β, it should increase the congestion window to keep the m e number of packets in the buffer when the propagations delay increases.

La et al. [8] proposed a modification to the Vegas to counteract the rerouting problem. Their modification uses any lasting increase in RTT as a sign of rerouting. For the first K packets, the mechanism is the same as TCP Vegas. However, subsequently, the source keeps track of the minimum RTT of every N packet. If this minimum RTT is much larger than the baseRTT for L consecutive times, the source updates its baseRTT to the minimum RTT of the last N packets and resets the cwnd based on this new baseRTT. Their reasoning for this modification is that since the increase in RTT forces the source to decrease the cwnd, the increase in RTT comes mostly from the propagation delay of the new route and not from the congestion. However, there are two more parameters δ and γ in this scheme that update the baseRTT, and finding the appropriate values for K, N, L, 6 and y remains an open problem.

(iii). Unfair treatment of old connection

As pointed out in [5], TCP Vegas' congestion control mechanism is inherently unfair to older connections. When a Vegas connection is established in uncongested networks, the baseRTT is likely to be close to the minimal RTT possible. When, later on, the network gets congested, the measured RTT rtt, increases and the ration $baseRTT_1/rtt_1$ decreases. Consider a new connection that is initiated after the network becomes congested. The value of $baseRTT_2$measured for this connection will be larger than $baseRTT_1$. Hence $baseRTT_2/rtt_2$ will be larger than $baseRTT_1/rtt_1$ since rtt_1 and rtt_2 are nearly equal. As derived in [5], window size that would bigger a reduction in congestion window size is given by:

$$cwnd > \frac{\beta}{1 - \frac{baseRTT}{rtt}}$$

Thus, for the older connection, the critical value of *cwnd* is smaller than that of the newer connection. Hence, the second connection can achieve higher bandwidth than the first (older) connection. Similarly, the critical value of congestion window that triggers an increase in congestion window is given by:

$$cwnd < \frac{\alpha}{1 - \frac{baseRTT}{rtt}}$$

Since the right hand side term is larger for the newer connection, it is more likely to be able to increase its congestion window than the older connection, and thus again get a more than fair share of the network bandwidth.

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. Consider a connection that starts when the network is congested Due to the queuing delay caused by the back log from other connections, the packets from the new connection experiences RTTs that are considerably larger than the actual propagation delay of the path. Due to this inaccurate estimation of the baseRTT, it will overestimate the possible window size (as pointed out by the critical window sizes that trigger increase/decrease of the cwnd). This scenario will repeat for each new connection, leading to persistent congestion. La et al. [SI 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. 191 also proposed augmenting Vegas with appropriate active queue management (AQM) such as Random Exponential Marking (REM) for avoiding the persistent congestion problem.

iv. Solutions through TCP PIPE:

IV. TCP VEGAS MODIFICATION – TCP PIPE

In this section a modification to TCP Vegas is proposed, named TCP PIPE. TCP Vegas has fixed values for α and β, set to 1 and 3 usually. Recall that Vegas strategy is to adjust the source's sending rate (congestion window) in an attempt to keep a small number of packets buffered in the routers along the path. Thus, when Vegas connections compete with Reno connections, the fairness problem is worse with larger router buffers. Since the average number of packets in the router buffer is to be kept within α and β, the main idea in TCP PIPE is that, rather than fixing α and β they be made adaptive. At the start of a connection, α is set to 1 and β to 3. These values however change dynamically depending on the network conditions. The slow start and congestion recovery algorithm of TCP PIPE are the same as that of Vegas. However, TCP PIPE uses the following congestion avoidance mechanism:

Terms used:

Tp = actual throughout at time *t*; *Tp'* = actual throughput at previous rtt. The definitions of expected_rate, actual_rate and diff are the same as those in Vegas.

If β>*diff*>α {

If Tp > Tp' {

Cwnd = *cwnd* + 1;

/* this indicates the better utilization of available bandwidth and the sending rate can be increased. Hence the throughput increases */

α = α + 1, β = β +1;

/* because the throughput increases, the value of *diff* also increases. The existing values of α and β may cause the problem for *cwnd* growth, so here the increase if α and β is necessary */

}

else if Tp ≤ Tp' {

/* if actual throughput time at t is less than actual throughput time at previous rtt then there is no deed to change cwnd, *α, β* values */

cwnd = *cwnd;*
α=α;
β=β;

}

else if diff ≤ *α*
{

if α>1 and Tp > Tp'
{

cwnd = *cwnd* + 1;

}

else if α>1 and Tp ≤ Tp'
{

cwnd = cwnd – 1;

α = α – 1;

β = β – 1:}

else if α = 1 {

cwnd = cwnd + 1;}

else if diff ≥ β {

cwnd = cwnd – 1;

α = α – 1;

β = β – 1;}

else {

cwnd = cwnd;

α = α;

β = β;
}

V. Simulation Results:

Some extensive simulations to compare the performance of TCP PIPE with Original Vegas implementation under various network conditions are conducted[10]. This section details the results obtained.

A. Rerouting

In this scenario, there is a Vegas connection (S1 – D1). The simulation takes place a change in the route by changing the RTT of the link connecting S1 to R1. The link S1 – R1 has a bandwidth of 1Mbps and initial RIT of 20ms. After 20 seconds of file transfer over the connection, the RTT of the link is changed to 200ms. The links Rl-R2 and R2D1 have bandwidth of 1Mbps each and RTT of 10ms for the entire duration of the connection. The simulation was run for 200 seconds so as to give the connection time to stabilize. Table 1 shows the comparison of average throughputs (in bits/sec) obtained in the two cases.

Table 1 throughput for rerouting conditions

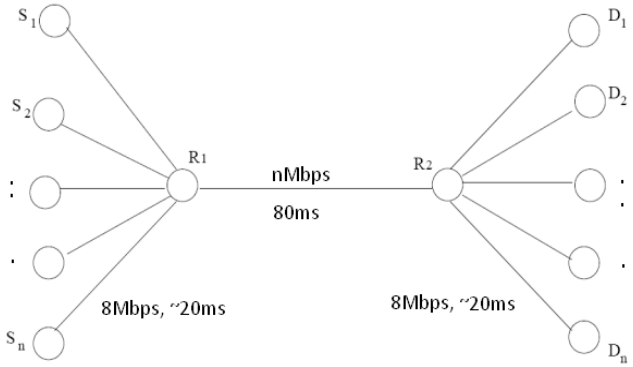
	TCP Vegas	TCP PIPE	Difference	% Increase
Throughput	217320	940240	722920	333

Figures 2 and 3 show the variation of throughputs for the full run. The dotted curve shows the instantaneous throughput, averaged intervals, and the solid one shows the average throughput as can be seen here, as soon as the RTT changes at 20 seconds, the instantaneous throughputs of Vegas and TCP PIPE start to decrease. However, the throughput of Vegas (see Figure 2) went all the way down to O.12Mbps. As seen in Figure 3, the throughput of TCP PIPE does suffer initially when the RTT changes at 20 seconds.

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. Then, when there is a slight improvement in the throughput for any particular RTT cycle, the TCP PIPE connection increases the value of mind, α and β.

This increase in *cwnd* in turn increases the throughput further, which will then trigger an increase in *cwnd*, α and β again. This process continues until *diff* becomes less than α Then *cwnd* alone increases. This growth of the congestion window is shown in Figure 5. The end result is that the bandwidth available is fully utilized and the throughput goes back

to the same large value prior to the rerouting. Figure 4 shows the throughput variation for TCP New Reno tinder similar conditions.



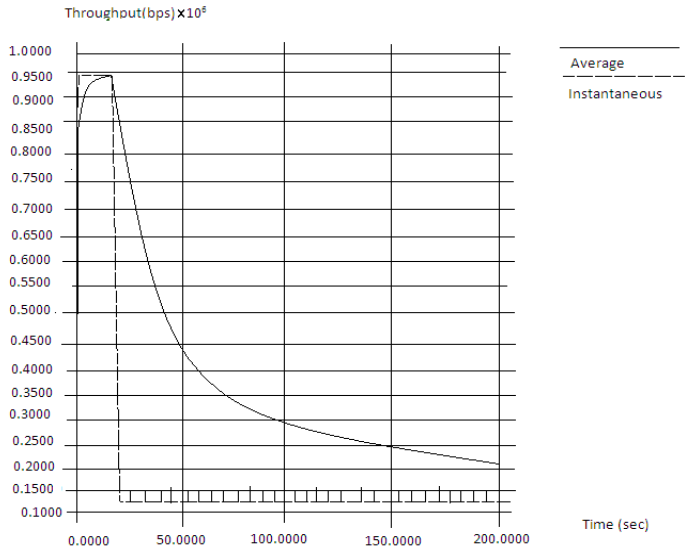


Fig1: Throughput variation of Vegas connection due to RTT change

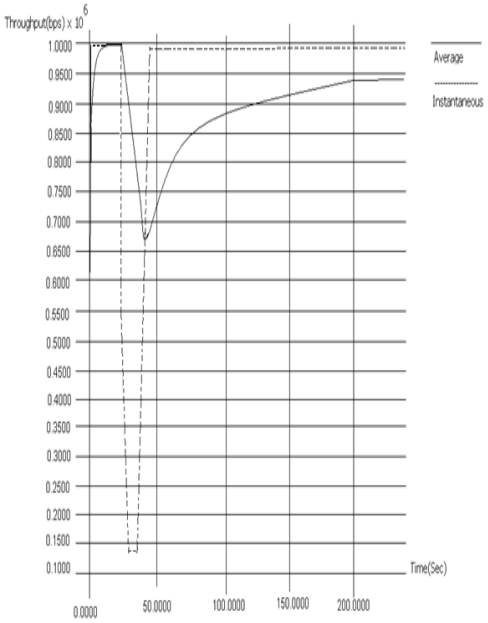


Fig2: Throughput variation of TCP PIPE connection due to RTT change

B. Bandwidth sharing with New Reno

Next, the performance of TCP PIPE when it sharer 'a link with TCP New Reno is considered. There are two connections, SI-Dl and S2D2. Both the sources are transferring files, but former one uses TCP PIPE or Vegas and the latter one uses TCP New Reno. The links S1- R1 and S2-R1 are both 8Mbps with RTT of 20ms. The bottleneck link R1-R2 is of bandwidth 800Kbps and RTT of 80ms. R2- D1 and R2-D2 links are both 8Mbps and RTT 20ms. S1 was started first and then S2's New Reno traffic was introduced after 10 seconds. Figure 6 shows the instantaneous and average throughputs of the TCP New Reno 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 New Reno starts at 10 seconds, it starts to consume Vegas' share of bandwidth.

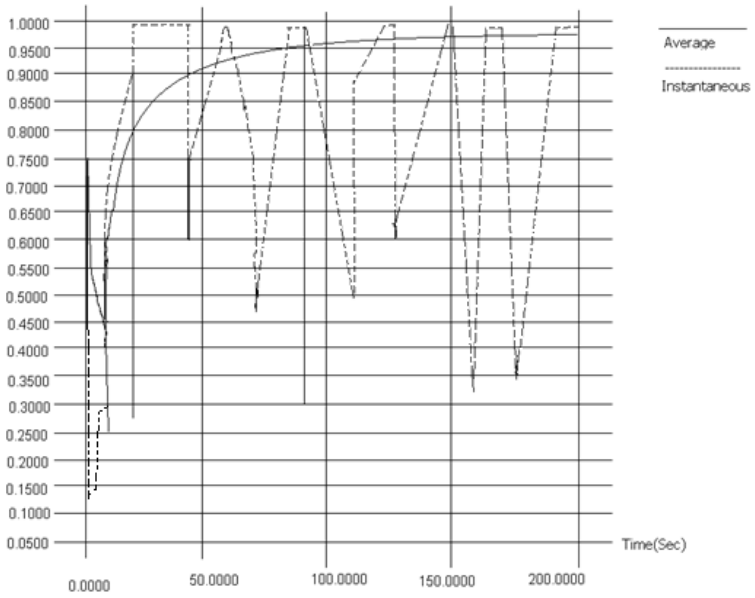


Fig3: Throughput variation of New Reno connection due to RTT change

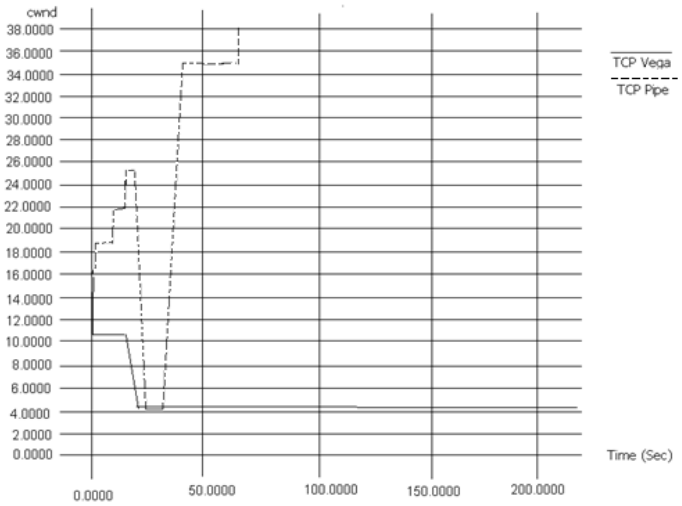


Fig4: cwnd variation for TCP Vegas and TCP PIPE connection due to RTT change

However, when TCP Vegas is replaced with TCP PIPE, the results are very different, as seen in Figure 6. Even though TCP New Reno gets a more than fair share of the bandwidth, the unfairness is not as much as in the case of TCP Vegas. Table 1 gives numerical value for the throughput of each connection. The use of TCP PIPE reduced the 'unfairness' from a ratio of 5.3:1 to 3.2:1.

Table2. Throughput ratios for New Reno-Vegas and New Reno-TCP PIPE connections

Vegas	TCP PIPE
132040 = 5.33	199320 = 3.17

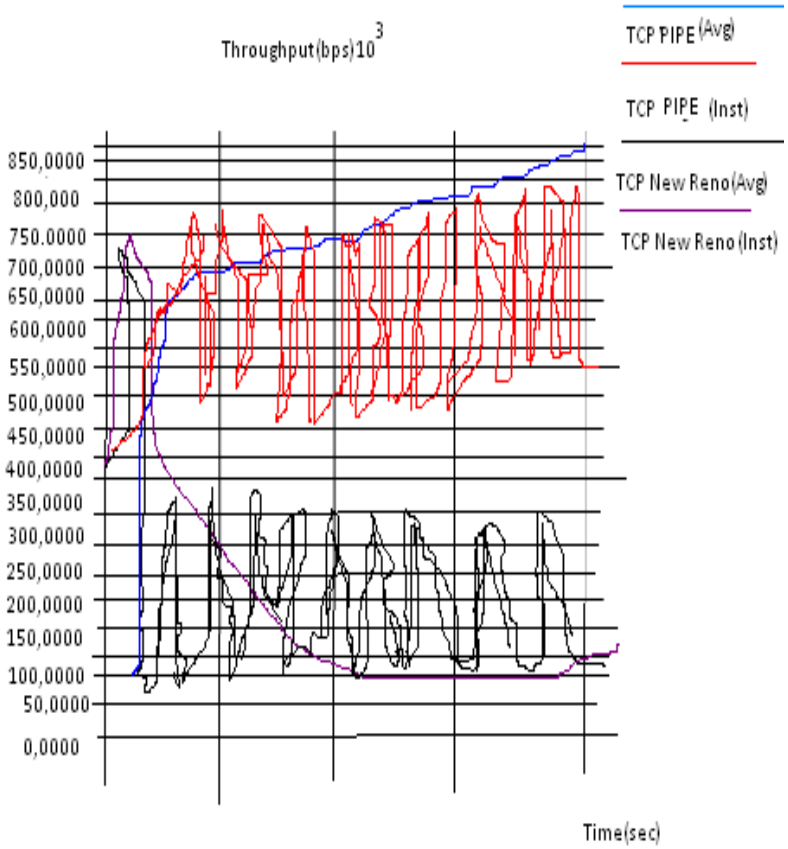


Fig5. Throughputs of TCP New Reno and TCP PIPE connections over congested link

When the above experiment was repeated with 3 New Reno sources and 3 Vegas/PIPE sources, TCP PIPE was found getting a more than fair share of the congested link bandwidth. Each source-to-R1 link was 1Mbps and RTT of 20ms. R1-R2 link was 1Mbps and RTT of 80ms. R2-todestination link was 1Mbps and had RTT of 20ms.

Figures 8 and 9 show the average throughputs obtained. The Vegas/PIPE sources (S1, S2, S3) were started at 0,10,20 seconds while the New Reno sources (S4, S5, S6) were started at 30,40,50 seconds. The average throughput obtained by each flow is given in Table 3.

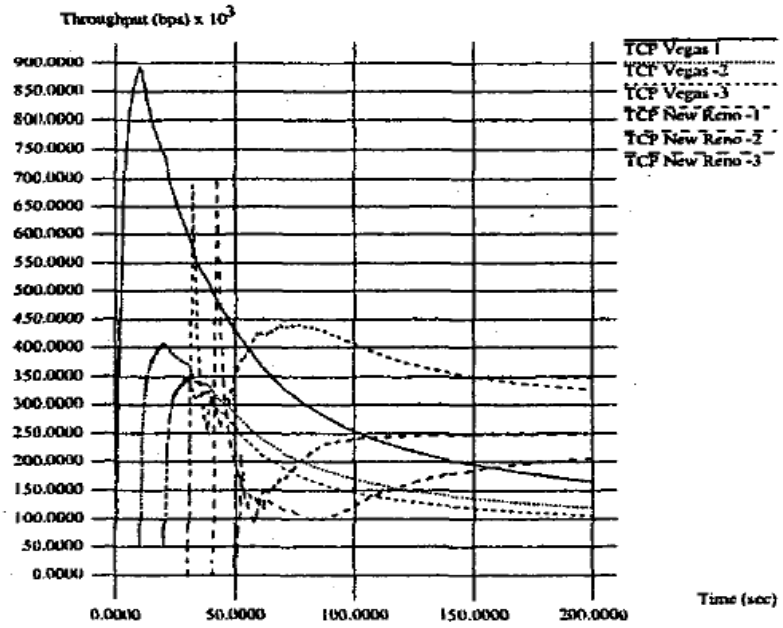


Fig6: throughput of 3 TCP New Reno and 3 Vegas connections over congested link

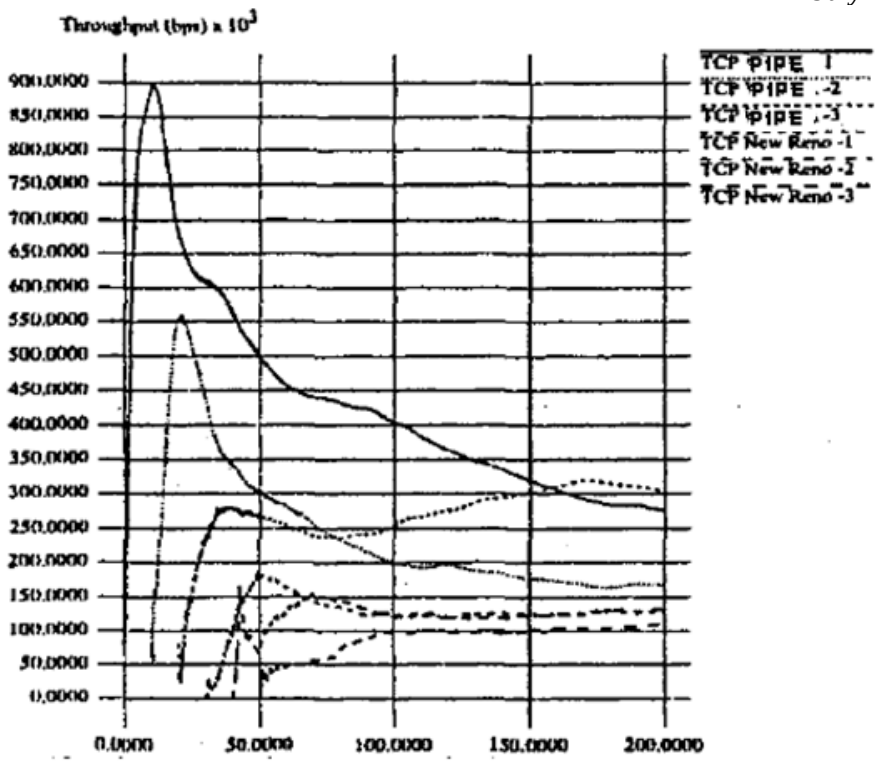


Fig7: Throughput of 3 TCP New Reno and 3 TCP PIPE connections over congested link

Table3: Throughputs of New Reno/Vegas and New Reno/ TCP PIPE connections

sources	Vegas & New Reno	TCP PIPE & New Reno
S1	164480	275480
S2	119242	167789
S3	106044	304178
S4	326590	127908
S5	207002	131402
S6	250828	109336

From the values in Table 3, the Internet protocol fairness ratio [I I] for the Vegas & New Reno connections can be calculated as 0.497 and that of the TCP PIPE & New Reno connections as 2.028. Inter protocol fairness ratio is the ratio of the average bandwidth shares between the two protocols, i.e., it is the ratio of average TCP Vegas/TCP PIPE bandwidth calculated across all the Vegas/PIPE flows to the average TCP New Reno bandwidth calculated across all the New Reno flows. The obtained values show that TCP PIPE actually outperforms New Reno.

C. Fairness between old and new connections

As mentioned in Section 3.C. TCP Vegas has the disadvantage that newer connection of Vegas enjoys a larger throughput because of the discrepancy in the estimation of base RTT. TCP PIPE's modified congestion control mechanism overcomes this shortcoming. To illustrate this, a simulated a scenario where 5 TCP Vegas/TCP PIPE connections are sharing a link is conducted. The source-to-router links were set to 1Mbps and RTT of 2Oms, while the router-to destination links had a bandwidth of 1Mbps and RTT of 100ms. The sources were started at intervals of 50 seconds each. Table 4 shows the average throughputs obtained by the connections for the simulation lasting 900 seconds. Figure 9 shows the average throughputs of the Vegas connections, while Figure 10 shows that of the TCP PIPE connections. Table4. Throughputs of five Vegas and TCP PIPE connections.

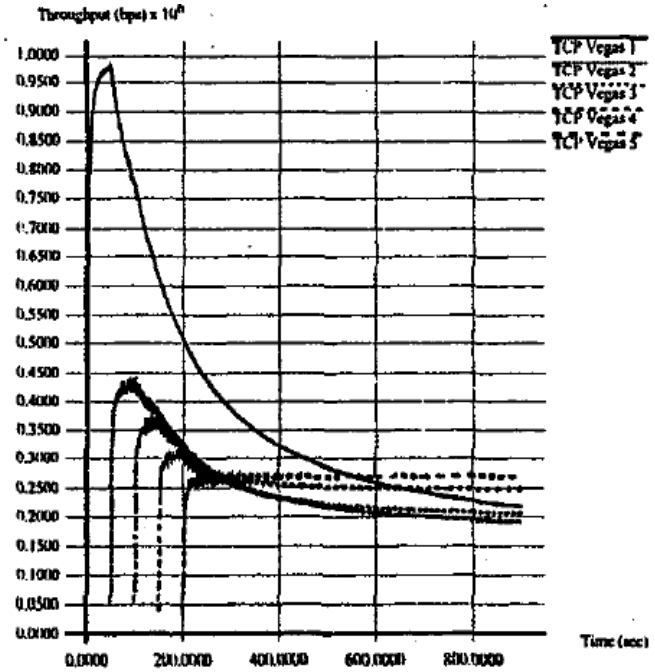


Fig8: Throughput of 5 Vegas connections over congested link

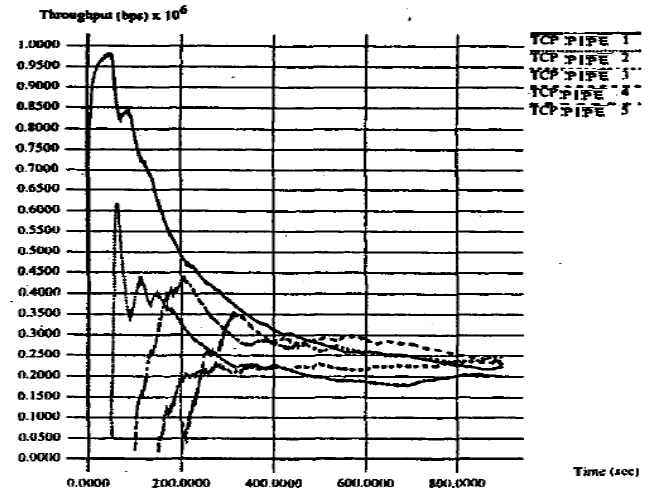


Fig9: Throughput of 5 TCP PIPE connections over congested link

D. Bias against high bandwidth flows

Hasegawa et al.[12] showed that TCP Vegas, like Reno, has the fairness bias against connections with higher bandwidth, to test whether TCP PIPE can perform better than Vegas in terms of this fairness, simulations were conducted with 3 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 was set to 5ms.R1-R2 link was bandwidth limited to 400kbps and the RTT was set to 10ms.R2-to-destination link had a bandwidth of 1mbps and RTT of 5ms.The simulations were carried out for a period of 1800 seconds. Table 5 summarizes 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.

Table 5.Comparison of Vegas and TCP-PIPE bias against high bandwidth connection for 1800 seconds

	S1	S2	S3
Expected	57.14	114.29	228.57
Vegas	123.34	146.85	120.46
TCP-PIPE	98.90	134.54	138.25

Ideally, S3-D3 connection should have received 228Kbps of the bottleneck bandwidth, but when Vegas is used, the connection ends up with just 120.5Kbps bandwidth. However, when the source is switched to TCP PIPE, the three connections enjoy fairer shares of the bandwidth (closer to the expected values). As observed, in the case of Vegas, the cwnd attains a steady value, in the very early stage of the connection, and remains like that for the entire duration of the connection. In the case of TCP-PIPE, α and β can vary dynamically, allowing cwnd to probe for more bandwidth share.

E. Retaining the properties of Vegas

TCP PIPE tries to improve upon the congestion control mechanism of TCP Vegas by trying to adapt to the availability of the network bandwidth. However, it is essential that in doing so, TCP PIPE should not lose the properties of Vegas. To ensure that TCP PIPE does inherit the properties of Vegas, simulations were conducted with only one Vegas/TCP PIPE/new Reno source connected through the routers to the destination. The S1-R1 link bandwidth was set to 1Mbps and the RTT of the link to 5ms and 45ms, respectively, for two sets of experiments. The RI-= bandwidth was restricted to 25OKbps and the RTT to 5ms and 45ms, respectively, for the two sets of experiments. The R2-D2 link bandwidth was set to 1mbps and RTT to 10ms. File sizes of 10MB and 5MB were sent from source to the destination. Tables 6 and 7 show 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 tables show, Vegas and TCP-PIPE outperform New Reno in all performance measures, as expected. The only difference between Vegas and TCP-PIPE is that the average buffer occupancy at the routers is larger for TCP-PIPE compared to Vegas. This is expected since TCP-PIPE adapts a and p to values larger than I and 3, which in turn let TCP-PIPE 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 does not seem to increase the time required to complete the transfer of the files, thus showing that the queuing delay is not increased much.

Table6.Comparison of New Reno, Vegas and TCP PIPE connection on a 20ms RTT link

Source	5MB file			100MB file		
	Time	Avg.Queue	Retx.pkts	Time	Avg.Queue	Retx.pkts
New Reno	162.353	11.46	58	322.353	23.27	62
Vegas	160.253	0.65	0	320.253	1.3	0
TCP-PIPE	160.253	5.62	0	320.256	12.4	0

Table7. Comparison of New Reno, Vegas and TCP-PIPE connections on a 100ms RTT connection

source	5MB file			10 MB file		
	Time	Avg.Queue	Retx.pkts	Time	Avg.Queue	Retx.pkts
New Reno	166.505	11.29	59	326.505	23.0	62
Vegas	160.651	0.82	0	320.651	1.63	0
TCP-PIPE	160.654	4.85	0	320.651	10.84	0

Next the performance of Vegas is studied TCP-PIPE and new Reno when the router queue sizes were varied. The RTT of the source to destination links were fixed at 40ms.S1-R1 and R2-D1 link bandwidths were set at 1Mbps, while the R1-R2 link bandwidth was set at 500kbps.Files of size 10Mb was sent from S1 to D1.Table 8 shows the values obtained. The unit of time is seconds and the retransmissions (Retx.) records the number of packets retransmitted.

Table 8. Comparison of New Reno, Vegas and TCP-PIPE connections with different router buffer queue size

Buffer size										
Source	Time	Retx.	Time	Retx.	Time	Retx.	Time	Retx.	Time	Retx.
New Reno	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
TCP-PIPE	160.5	2	160.6	1	160.6	1	160.4	0	160.4	0

The results show that when the buffer size is as small as 10,15 or 20, TCP PIPE retransmits at the most, 1 or 2 packets, while Vegas does not lose any packets at all. This number is extremely small compared to the number of packets dropped

and retransmitted when New Reno is used. Furthermore, when the queue size is increased to 25 and 30, the behavior of TCP PIPE approaches that of Vegas.

VI. Conclusion

In this paper, a modified version of TCP Vegas I introduced, TCP PIPE, which is able to mitigate some of the limitations of TCP-PIPE performs better when competing with other TCP connections like New Reno for shared bandwidth; TCP PIPE overcomes the routing limitations of Vegas and is able to adapt to the changes in RTT and routers faster; TCP PIPE connections do not suffer from the 'unfairness towards old connection' and 'unfairness against higher bandwidth connections' problems of Vegas. It was also shown that even though TCP PIPE is different from Vegas, the basic congestion control algorithm still remains as effective as Vegas in decreasing the average queue occupancy and packet retransmissions.

References:

[1] L.S. Brakmo, S.W. OMalley, and L.L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance,"Proc. ofACMSIGCOMM94, pp. 24-35. Octobcr 1994.
[2] L.S. Brakmo and L.L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet," IEEE Journal on Selecred Areas in Conrniunimrions. I3(8): 1465-80, October1995.
[3] J.S. Ahn, P. Danzig. z. Liu, and L. Yan. "Evaluation of TCP Vegas: Emulation and Experiment," in Proc. of ACM SIGCOMM'95, pp. 185-195, Allgust 1995.
[4] Jeonghoon MO, Richard J. La, Venkat Anantharam, and Jean Walrand, "Analysis and comparison of TCP Reno and Vegas," in Proc. of IEEE INFOCOMM99. pp. 1556-1563, March 1999.
[5] U. Henpartner. I. Bolliger. and Th. Gross. "TCP Vegas Revisited," in Prm. of IEEE INFOCOM '2000, pp. 1546-1555, March 2000.
[6] Hasegawa, K. Kurata, and M. Murata, "Analysis and Improvement of Fairness between TCP Reno and Vegas for Deployment of TCP Vegas to the Internet," in Pm. of he IEEE International Confrence on Nehvurk Pmrocols (ICNP 2WO). November 2000.
[7] Raghavendra and R.R Kinicki, "A Simulation Performance study of TCP Vegas and Random Early Detection," in Pm. Of IPCCC'99, pp. 169-176, February 1999.
[8] Richard j. La, Jean Walrand, and Venkat Anantharam, "Issues in TCP Vegas," available at http://w.path. bcrkelq.edd-hyongla, July 1998.
[9] Steven Low, Lany Peterson, and Limin Wang, "Understanding TCP Vegas: A Duality Model." in Pruc U/ ACMSIGME7WCS2OOI, pp. 226235, June 2001.
[10] Network Simulator (NS), http://www.isi.edu/nsnam/ns
[11] Rem Rcjaie, Mark Handley, and Deborah Estrin, "RAP: An end-toad rate-based congestion control mechanism for realtime streams in the Internet," in Proc. of IEEE INFOCOM99, pp.1337- 1345, Mmh 1999.
[12] Hasegawa T. Mat.w M. Muram and H. Miyahara, "Comparisons of Packet Scheduling Algorithms for Fair Service among Connections on the Internet," in Prac. of IEEE INFOCOM 2000,pp.1253-1262,2000.

Author(s):

Dr.R.Seshadri Working as Professor & Director, University Computer Centre, Sri Venkateswara University, Tirupati. He was completed his PhD in S.V.University in 1998 in the field of " Simulation Modeling & Compression of E.C.G. Data Signals (Data compression Techniques) Electronics & Communication Engg.". He has richest of knowledge in Research field, he is guiding 10 Ph.D in Fulltime as well as Part time. He has vast experience in teaching of 26 years. He published 10 national and international conferences and 15 papers published different Journals.

Prof.N.Penchalaiah Research Scholar in SV University, Tirupati and Working as Professor in CSE Dept,ASCET,Gudur. He was completed his M.Tech in Sathyabama University in 2006. He has 13 years of teaching experience. He guided PG & UG Projects. He published 2 National Conferences and 9 Inter National Journals.

