A Simulation Study on the Throughput Fairness of TCP Vegas[1]

Estella C. M. Tsang and Rocky K. C. Chang

Department of Computing
The Hong Kong Polytechnic University
Hung Hom, Kowloon, Hong Kong
Email: csrchang@comp.polyu.edu.hk

Abstract

Throughput fairness is an important criteria for evaluating TCP performance. Fairness is especially important for best effort service, which is still the dominant type of service in the Internet and, predictably, in the years to come. However, the TCP protocols prevailing in the Internet, including TCP Tahoe and TCP Reno, are known to be unfair, especially to connections with larger round-trip delays. Using the ns simulator, we have thoroughly examined the fairness of TCP Vegas focusing on three issues: 1) Is TCP Vegas really fair to connections with larger propagation delays? 2) what is the impact of the thresholds, α and β, used in TCP Vegas' congestion avoidance algorithm on fairness? and 3) when there is a mixture of TCP Vegas and TCP Reno connections, are TCP Vegas and TCP Reno fair to each other? The simulation results support that TCP Vegas is still unfair to connections with larger propagation delays, for example, when $\alpha = 1$ and $\beta = 3$. But, unlike TCP Reno, the delay bias does not necessarily increase as the delay difference increases. The unfairness problem can be resolved by an Enhanced TCP Vegas that sets $\alpha = \beta = 2$ or 3 but not 1. When $\alpha = \beta = 1$, the fairness is unstable and may be worse than that when $\alpha = 1$ and $\beta = 3$. Considering a trade-off among fairness, stability and aggressiveness, a value of 3 seems to be an acceptably good choice. Finally, fairness between TCP Reno and TCP Vegas connections depends on the RED gateway thresholds, the number of active flows, and TCP Vegas parameters α and β.

1. Introduction

Throughput fairness is an important criteria for evaluating TCP performance. In this paper we will adopt Jain's fairness index [8] to evaluate the throughout fairness among senders with different round-trip times. Given a set of flow throughputs (b_1, b_2, ..., b_n), the fairness index is defined as:

$$\left(\sum_{i=1}^{n} b_i\right)^2 \Big/ n \times \left(\sum_{i=1}^{n} b_i^2\right).$$

Since throughputs are nonnegative quantities, the fairness index is always between 0 and 1. A lower value implies poorer fairness. If all throughputs are the same, the fairness index is 1. If a link is shared equally among any k of n senders (others zero), the Jain's fairness index is equal to k/n. If the bottleneck link is shared by two connections and the Jain's index is 0.5, then one of them is effectively shut out. If the fairness index is 0.9, then one of the connections is getting 2/3 of the bandwidth. Therefore, a fairness index of less than 0.9 should not be considered a good value.

Fairness is especially important for best effort service, which is still the dominant type of service in the Internet and, predictably, in the years to come. However, the TCP protocols now prevailing in the Internet, including TCP Tahoe and TCP Reno, are widely known to be unfair, especially to connections with larger round-trip delays [5, 9]. Floyd and Jacobson suggested that TCP Reno's bias against longer round-trip time connections was a consequence of TCP's own window increase algorithm [5]. Moreover, they showed that the bias was unrelated to the gateway dropping policy, because both the Drop Tail (FIFO) and the Random Drop gateways shared such bias. In their simulations, the improvement was minor even when the RED gateways were used.

In this paper we study the fairness of TCP Vegas, which was proposed by Brakmo and Peterson [3]. There have been some studies to show that Vegas fairness is better than Reno fairness [4, 11] and that TCP Vegas does not suffer from such delay bias as TCP Reno does. Mo et al. used a simple fluid model to show that TCP Vegas did not suffer from delay bias as TCP Reno did and validated their analysis by simulations [11]. For a network topology consisting of 2 connections sharing a bottleneck link, their result demonstrated that the throughput ratio of TCP Vegas ranged from 1.09 to 1.33 whereas the throughput ratio of TCP Reno ranged from 1.35 to 37.12. Moreover, the simulations showed that the throughput ratio of TCP Reno was increased when the difference between the two connections' propagation delays increased. In contrast, this phenomenon did not happen with TCP Vegas.

Hasegawa et al. also pointed out that the difference in the values of α and β caused an unfair bandwidth sharing when connections running TCP Vegas shared a bottleneck link [7]. They argued that for $\alpha = 1$ and $\beta = 3$, TCP Vegas might occupy between 1 to 3 buffers in the intermediate routers.

[1] This work is partially supported by The Hong Kong Polytechnic University Research Grant S909.

Therefore, in the worst-case scenario, one connection might occupy about 3 router buffers while the other might only occupy 1 router buffer. Consequently, one connection was able to achieve 300% bandwidth of the other. This led to proposals that the values of α and β should be the same, and the resulting TCP Vegas was referred to as Enhanced TCP Vegas. They claimed that Enhanced TCP Vegas provided better fairness by removing the condition that allowed the window size to stay fixed as long as the difference between the expected sending rate and the actual sending rate lied between a narrow range of values.

Other previous work demonstrated that unfairness existed when TCP Vegas competed for bandwidth with TCP Reno [1, 11, 12]. Moreover, the unfairness occurred because TCP Vegas used a more conservative congestion avoidance mechanism. The explanation is as follows. When a TCP Vegas connection shares a link with a TCP Reno connection, the TCP Reno connection uses most of the buffer space while the TCP Vegas connection, interpreting this phenomenon as a sign of congestion, continues to back off. Therefore, TCP Reno is able to capture a higher throughput. When the buffer is small, however, TCP Vegas outperforms TCP Reno. Mo et al. [11] provided an intuitive explanation: TCP Reno needs some room in the router buffer for oscillations in order to estimate the available bandwidth. Without the necessary room in the buffer, its performance degrades. TCP Vegas, on the other hand, quickly adapts to the small buffer sizes since it requires only a few buffers. They simulated a network with 2 connections having the same propagation delay, and demonstrated that the throughput ratio of Reno to Vegas increased from 0.535 to 11.73 when the router buffer increased from 4 to 50.

This paper concentrates on three fairness issues: 1) Is TCP Vegas really fair to connections with larger propagation delays? 2) what is the impact of the thresholds, α and β, used in TCP Vegas' congestion avoidance algorithm on fairness? and 3) when there is a mixture of TCP Vegas and TCP Reno connections, are TCP Vegas and TCP Reno fair to each other? For all issues, we summarize the differences between our approaches and results, and that in the previous studies in Tables 1-2.

2. TCP Vegas

TCP Vegas was based on modifications to TCP Reno, and the three major modifications are an extension to Reno fast retransmission algorithm, a modified congestion avoidance algorithm, and a modified slow-start. There are also other minor modifications [6]. However, only the modified congestion avoidance algorithm has major impact on the fairness issue. Therefore, we will only discuss the modified congestion avoidance algorithm in this section.

2.1 Modified congestion avoidance algorithm

Instead of increasing the congestion window size until losses occur as TCP Reno does, a TCP Vegas sender tracks the changes in the throughput (or more specifically, changes in the sending rates) and then adjusts the congestion window size. It observes changes in the round-trip times of the segments that the connection has sent before. It then calculates and compares the measured throughput against the expected throughput. If the expected sending rate is higher than the actual sending rate by α or less, the TCP Vegas sender thus increases the congestion window by one. If the expected rate is higher than the actual rate by β or more, it assumes that congestion starts to build up and thus decreases the congestion window by one. Otherwise, the congestion window remains unchanged. TCP Vegas makes this calculation and decision in window adjustment once per RTT. The increment, however, is made on receipt of each acknowledgement, resulting in an increase of about 1 per RTT. The new algorithm is summarized below:

During the congestion avoidance phase, a TCP Vegas sender does :

cwnd = cwnd + 1 if diff < (α/baseRTT)
cwnd = cwnd if (α/baseRTT) $\leq$ diff $\leq$ (β/baseRTT)
cwnd = cwnd – 1 if (β/baseRTT) < diff,
where

- diff = expected rate – actual rate ≥ 0, by definition,
- expected rate = data in transit/baseRTT,
- baseRTT = the minimum of all measured RTTs, typically the RTT of a packet when the router queue is empty or when the flow is not congested (in seconds),
- actual rate = (next send sequence number – segment timed)/average RTT,
- RTT = observed or actual round trip time (in seconds),
- α, β = some constant thresholds.

The objective of TCP Vegas' congestion avoidance mechanism is to measure and control the amount of extra data in the network. These extra data inevitably has to be buffered in the router. It can be shown that, in the absence of other connections, the congestion window of TCP Vegas converges to a fixed value with between α to β extra bytes in the network.

In comparison, TCP Tahoe and TCP Reno perform congestion control, i.e. they control congestion once it happens. They treat packet losses as a signal of congestion. Therefore, they consistently cause packet losses to themselves and to other connections, resulting in under-utilization of the bottleneck link. TCP Vegas, on the other hand, performs congestion avoidance from end nodes (i.e. no router involvement is expected). It monitors changes in the sending rate (and RTTs) to predict congestion before losses occur.

2.2 Enhanced TCP Vegas

Hasegawa, Murata, and Miyahara proposed Enhanced TCP Vegas to solve the unfairness problem that they observed from TCP Vegas. The major difference is that the control thresholds, α and β, are set to the same value δ in Enhanced TCP Vegas. The resulting congestion avoidance mechanism is summarized below. Other than this, Enhanced TCP Vegas uses the same aggressive retransmission strategy and conservative slow-start strategy as in TCP Vegas.

During the congestion avoidance phase, Enhanced TCP Vegas does :

cwnd = cwnd + 1 if diff < (δ/base_RTT)

cwnd = cwnd – 1 if diff ≥ (δ/base_RTT).

3. Simulation Results

3.1 Network model

The network topology, as shown in Fig. 1, consists of two sources (S1 & S2), two destinations (D1 & D2), and two routers (R1 & R2). Connection 1 starts from S1 to D1 and connection 2 starts from S2 to D2. Both connections are established via R1 and R2. The link between R1 and R2 is shared between the 2 connections. The bandwidth of the shared link is 1.5Mbps. The buffer size of router R1 is 100 packets. The round-trip propagation delay between S1 and D1 (connection 1), the shorter link, is 6ms by setting x1 to 1ms. Thus, the minimum round-trip time of the shorter link, including processing delay of the packets in the intermediate routers, is 13.211ms. The round-trip propagation delay between S2 and D2 (connection 2), the longer link, is varied by changing the value of x2. For example, if x2 = 30ms, the round-trip propagation delay of connection 2 is 64ms. Therefore, the minimum round-trip time of the longer path, including the transmission delay at the intermediate routers along the path, is 71.211ms.

The routers are assumed to be RED routers for which the RED minimum and maximum thresholds are 15 and 45, respectively, unless otherwise stated. Larger thresholds (instead of the typical values of 5 and 15, respectively) are chosen to ensure that the limit on the average queue length cannot affect the TCP senders' performance.

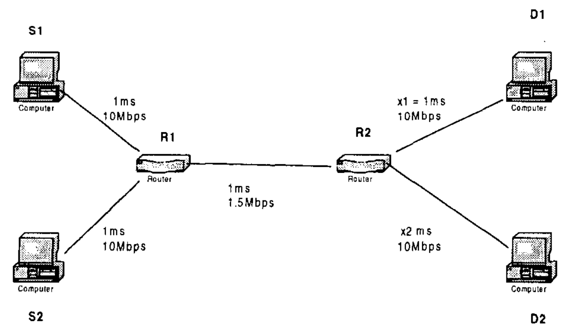

Fig. 1. The simulation network model.

The maximum window size that a TCP connection can reach will have an impact on its performance. The window size is therefore set to 30 in our simulations. Moreover, the maximum value of the minimum round-trip time (including the processing delay) is capped to ensure that the window size does not pose a limit on the TCP performance. For TCP Vegas, α and β are set to 1 and 3, respectively (this particular case is referred to as Vegas1_3).

Both connections start their own FTP transfers simultaneously. The sources are greedy sources such that they always have data to send. The packet size is 1000 bytes. Hence, the minimum round-trip time of connection 1, the shorter link, is equivalent to 2.48 packets. When x2 = 30ms, the minimum round-trip time of connection 2 is equivalent to 13.35 packets. If the bandwidth is equally shared among both connections and the bandwidth is fully utilized, then connection 1 should maintain on average about 2.24 to 4.24 packets in the network, while connection 2 should maintain on average about 7.67 to 9.67 packets in the network.

We use ns simulator [10] for all simulation experiments. Each simulation experiment lasts for 60 seconds, which is considered adequate, because TCP Vegas congestion window normally converges in less than 5 seconds. The average throughputs (or the average sending rates of the sources) of both connections are observed after the simulation has started for 10 ms.

3.2 Unfairness of TCP Vegas (when $\alpha \neq \beta$)

We have performed simulations with two TCP Vegas connections. The propagation delay of the shorter connection (connection 1) is set to 6ms, and that of the longer connection (connection 2) is ranged from x2 = 1ms to x2 =110ms (in a step of 10ms). There are no packet losses recorded in all the simulations runs.

The simulation results in Table 3 suggest that there is a bias against connections with larger propagation delays when $\alpha = 1$ and $\beta = 3$. The converged actual sending rate of the shorter connection is always higher than that of the longer connection, unless the difference in the propagation delays between the two connections is small (i.e. when x2 = 1ms or 5ms). When the propagation delays of both connections are close, the fairness index is equal to 1. In other settings, the fairness index varies from 0.90 to 0.996, and the converged actual sending rate of the shorter connection is always higher.

However, the degree of bias does not necessarily increase as the propagation delay of the longer connection increases. For instance, the fairness is worst when x2 =10ms, with the shorter connection transmitting at twice the rate of the longer connection.

Here we use the model developed in [2] to explain the unfairness for this case. Consider 2 users sharing a single bottleneck link. The propagation delay of connection *j* is d_j. The bandwidth of the bottleneck link is *c*. The minimum round-trip delay and the measured round-trip delay of connection *j* are baseRTT_j and RTT_j, respectively. Assume that the congestion window converges to a steady state value. Denote the difference between the expected rate and the sending rate by α_j (in packets), i.e. α_j = (expected rate – actual rate) * baseRTT_j.

Then, the minimum and the measured round-trip delay can be expressed as $\text{baseRTT}_j = d_j + x_j$, where x_j = an over-estimation of the propagation delay by connection j, and $\text{RTT}_j = d_j + \Delta_j$, where Δ_j = queuing delay experienced by connection j. At steady state, $cwnd_j - (baseRTT_j / RTT_j) * cwnd_j = \alpha_j$, where $\alpha \leq \alpha_j \leq \beta$ or $cwnd_j = \alpha_j * (d_j + \Delta_j) / (\Delta_j - x_j)$. Hence, the expression for connection j's actual throughput is given by $rate_j = cwnd_j / RTT_j = \alpha_j / (\Delta_j - x_j)$.

Therefore, one of the causes contributing to the unfairness is the difference in α_j of the two connections, even when x_j is zero, i.e. when there is no over-estimation of the propagation delay. In particular, α_j is allowed to converge to any value between α and β. In addition, α_j represents the number of packets queued at the router buffer when the steady state is reached. According to the FIFO policy, the ratio of buffer occupancy in the FIFO gateway reflects the ratio of the bandwidth sharing. If α_js of the two connections are different, then TCP Vegas will not achieve fairness.

3.3 Fairness of Enhanced TCP Vegas (when $\alpha = \beta$)

In this section, we study the fairness of Enhanced TCP Vegas' fairness which is labeled by Vegas-δ, e.g., Vegas-2 has $\delta = 2$. We also compare their fairness performance with Vegas-1_3.

The setting is similar to that stated in our network model. We show the simulation results for $\delta = 1$, 2, and 3 in Tables 4-5.

For Vegas-1, the actual sending rate of the longer connection fluctuates within a narrow range around the lowest edge of the fluctuation region of the actual sending rate of the shorter connection (the figure is not shown here). In addition, the range of the actual sending rate of the longer connection is very small. Unlike the longer connection, the actual sending rate of the shorter connection fluctuates within a very wide range from about 60 packets per second to 288 packets per seconds. Hence, the shorter connection is sometimes able to capture much higher goodput. Therefore, the average goodput of the shorter connection is also higher. In comparison, the actual sending rate of the shorter connection fluctuates between much narrower ranges with Vegas-2 (90 – 125 packets per second) and Vegas-3 (78 – 117 packets per second), thus leading to an improvement in fairness.

Moreover, the expected sending rate of the longer connection of Vegas-1 fluctuates around 60 packets per second (or about 480,000 bps), a value that is even less than its fair share. This means that the actual sending rate of the longer connection of Vegas-1 will be less than the fair share because the actual sending rate is less than the expected sending rate. The problem is much relieved with Vegas-2 and Vegas-3.

The simulation results suggest that $\delta = 1$ may not be large enough for a connection with large propagation delay to compete for a fair share of the available bandwidth. Of course, there is another limiting factor, i.e. the advertised window. The advertised window will limit the throughput that the longer connection can achieve.

In summary, the fairness is generally better for a large δ. However, there is a tradeoff: with a larger δ, TCP Vegas tries to force more extra packets into the network. Therefore, the larger the δ, the more aggressive TCP Vegas is and thus raises the chance of causing congestion in case of limited available bandwidth. From the simulation results obtained for this study, $\delta = 3$ seems to be an acceptably good value to use.

3.4 A mixture of Vegas and Reno connections

We consider a mixture of TCP Vegas and TCP Reno connections in this section: 30%, 50%, and 70% of the connections are TCP Vegas. Two Vegas configurations are compared : 1) $\alpha = \beta = 3$ and 2) $\alpha = 1$ and $\beta = 3$. Table 6 below presents the fairness indices with 2 settings of RED thresholds: 1) RED-5-15 : when the RED minimum threshold is 5 and the maximum threshold is 15, or 2) RED-54-108 : when the RED minimum threshold is 54 and the maximum threshold is 108. The fairness measures are the averages over seven simulation runs. The figures next to the fairness measures give the average goodput of all the TCP Vegas connections and that of all the TCP Reno connections, respectively. For instance, with RED-54-108, $\alpha = \beta = 3$ and when 10% of the 200 connections are Vegas connections, the fairness index is 0.919. The corresponding average goodput of all the Vegas connections is 67,306 bps while the average goodput of all the Reno connections is 47,123 bps.

The simulation results show that, when the RED thresholds are high and the number of active flows is low (30 when $\alpha = \beta = 3$, and 30 or 100 when $\alpha = 1$ and $\beta = 3$), TCP Reno is the winner. However, when the number of active flows is large (100 or 200 when $\alpha = \beta = 3$, and 200 when $\alpha = 1$ and $\beta = 3$), TCP Vegas becomes the beneficiary of the unfairness problem.

When the RED thresholds are low and the number of active flows is relatively low (e.g. 30 when $\alpha = \beta = 3$, and 100 when $\alpha = 1$ and $\beta = 3$), even each TCP Vegas connection may capture higher goodput than the TCP Reno connections on average. Intuitively, this suggests that when the RED thresholds are high enough, TCP Reno is able to benefit from a larger window and thus becomes the winner in the unfairness problem.

Due to the setting of the thresholds, the RED-5-15 gateway starts to drop packets when its queue is longer than 5 packets and drops all arrived packets when its queue is longer than 15 packets. When there are 100 connections, it is highly likely that these queue lengths are met and thus there will be a large number of packet drops. The retransmission of dropped packets is unavoidable. When there are too many losses, timeouts are also inevitable.

4. Conclusions and Future Work

We have presented in this paper part of the simulation results obtained for evaluating fairness of TCP Vegas. In the forthcoming paper, we will also report the results for many-flow scenarios. For example, when all connections are running either TCP Reno or TCP Vegas, the fairness of TCP Reno is better and more stable than that of TCP Vegas. The contrary occurs when the number of flows times the threshold of Enhanced TCP Vegas is less than the RED minimum threshold. Intuitively, this suggests that the fairness of TCP Vegas is better when there are few or no retransmissions or timeouts. Moreover, the fairness of TCP Vegas host-only configuration improves when the RED thresholds are higher. The percentage improvement, however, diminishes when the number of flows is large enough.

References

[1] Ahn, J. S., Danzig, P. B., Liu, Z., and Yan, L. "Evaluation of TCP Vegas: Emulation and Experiment," *Proc. ACM SIGCOMM*, pp. 185-195, Oct. 1995.

[2] Boutremans, Catherine. and Boudec, Jean-Yves Le Boudec. "A Note on the Fairness of TCP Vegas," *Proc. Intl. Zurich Seminar on Broadband Commun.*, pp. 163-170, 2000.

[3] Brakmo, L.S. and Peterson, L.L. "TCP Vegas: End to End Congestion Avoidance on a Global Internet," *IEEE J. Sel. Areas Commun.*, vol. 13, no. 8, pp. 1465-1480, Oct. 1995.

[4] Chen, J. R. and Chen, Y. C. "Vegas Plus: Improving the Service Fairness," *IEEE Communication Letters*, vol. 4, no. 5, pp. 176-178, May 2000.

[5] Floyd, S. and Jacobson, V. "Traffic Phase Effects in Packet-Switched Gateways," *ACM Computer Communication Review*, vol. 21, no. 2, pp. 26–42, Apr. 1991.

[6] Hengartner, U., Bolliger, J. and Gross, Th. "TCP Vegas Revisited," *Proc. IEEE INFOCOM*, pp. 1546-1555, Mar. 2000.

[7] Hasegawa, G., Murata, M., and Miyahara, H. "Fairness and Stability of Congestion Control Mechanisms of TCP," *Proc. IEEE INFOCOM*, pp. 1329-1336, Mar. 1999.

[8] Jain, R. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling.* John Wiley & Sons, Inc., 1991.

[9] Lakshman, T. V. and Madhow, U. "The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss". *IEEE/ACM Trans. Networking*, vol. 5, no. 3, pp. 336-350, June 1997.

[10] NS Network Simulator. Available from http://www-mash.cs.berkeley.edu/ns.

[11] Mo, J., La, R. J., Anantharam, V. and Walrand, J. "Analysis and Comparison of TCP Reno and Vegas," *Proc. IEEE INFOCOM*, pp. 1556-1563, June 1999.

[12] Raghavendra, Aditya M. and Kinicki, Robert E. "A Simulation Performance Study of TCP Vegas and Random Early Detection," *Proc. IEEE Intl. Computing and Communications Conference on Performance*, pp. 169-176, 1999.

Previous approaches (from [7] and [11])	Our approaches
1. Cover TCP Vegas only 2. Heuristic analysis and simulation	1. Cover Enhanced TCP Vegas as well 2. Extensive simulation study
3. Compare Enhanced TCP Vegas against TCP Vegas 4. Mainly through analytical analysis with limited simulation data as illustrative examples to support the argument	3. A complementary method to evaluate the fairness of Enhanced TCP Vegas & TCP Vegas through an extensive simulation study
5. The second connection starts after the 1st one has started for some time. a. v_baseRTT_ of 2nd connection may include propagation delay + transmission delay at router => higher throughput.	4. Amend ns to trace v_baseRTT_ (& other var.). 5. The results are free from the impact of an incorrect min. RTT measurement.

Table 1. A comparison of our approaches and the approaches adopted in previous studies.

Previous findings	Our findings
1. In the heterogeneous case (a mixture of TCP Vegas and TCP Reno connections), TCP Vegas does not suffer from delay bias as TCP Reno does. a. Delay bias of TCP Reno becomes very noticeable when the delay difference becomes large, but TCP Vegas does not show such pattern.	1. TCP Vegas (α=1 & β=3) still biases against connections with larger propagation delays, unless the difference in propagation delays is small. a. In some cases, the expected sending rate of the longer connection is less than the fair share of the available bandwidth.
2. In the heterogeneous case, TCP Vegas can improve fairness to some extent, but unfairness still exists.	2. Even in the heterogeneous case, the fairness of TCP Vegas may be worse than that of TCP Reno.
3. The Enhanced TCP Vegas can achieve good fairness at the expense of stability.	3. Support that the Enhanced TCP Vegas can solve the unfairness problem, but not when α=β=1.

Table 2. A comparison of our findings and the previous findings.

α=1 β=3	Connection 1: Propagation delay=6ms, Minimum round-trip time = 0.013211 seconds		Connection 2: Propagation delay = 2 * (x2 + 2)		Ratio of sending rates (Vegas1/ Vegas2)	Fair-ness Index
x2	Converg. expected sending rate	Converg. actual sending rate	Converg. expected sending rate	Converg. actual sending rate		
110ms	2422285	928571	642586	606382	1.53	0.965
100ms	2422285	925824	591103	545454	1.70	0.931
90ms	2422285	936107	610550	562500	1.66	0.941
80ms	2422285	926507	636363	583333	1.59	0.953
70ms	2422285	857142	724137	642623	1.33	0.980
60ms	2422285	934407	644480	568965	1.64	0.945
50ms	2422285	903420	686436	600000	1.51	0.962
40ms	2422285	928996	670091	571428	1.63	0.946
30ms	2422285	881562	731605	617647	1.43	0.970
20ms	2422285	800000	990379	700000	1.14	0.996
10ms	2422285	1000000	755794	500000	2.00	0.900
5ms	1816713	750000	1340183	750000	1.00	1.000
1ms	1816713	750000	1816713	750000	1.00	1.000

Table 3. Two Vegas1_3 connections (α = 1 and β = 3)– Fairness vs. connection 2's propagation delay (Min. RTT of the shorter connection = 0.013211s).

Min. RTT of the longer connection	Vegas-1	Vegas-2	Vegas-3	Vegas1_3
0.231 s. (x2=110ms)	0.928	0.995	0.992	0.965
0.211 s. (x2=100ms)	0.898	1.000	0.997	0.931
0.191 s. (x2=90ms)	0.890	1.000	0.996	0.941
0.171 s. (x2=80ms)	0.858	0.999	0.995	0.953
0.151 s. (x2=70ms)	0.988	0.982	0.989	0.980
0.131 s. (x2=60ms)	0.998	0.997	0.997	0.945
0.111 s. (x2=50ms)	0.997	0.986	0.996	0.962
0.091 s. (x2=40ms)	1.000	0.994	0.995	0.946
0.071 s. (x2=30ms)	0.997	0.968	0.991	0.970
0.051 s. (x2=20ms)	0.998	0.998	0.998	0.996
0.031 s. (x2=10ms)	0.988	0.999	1.000	0.900
0.021 s. (x2=5ms)	0.994	0.999	0.999	1.000
0.013 s. (x2=1ms)	1.000	1.000	1.000	1.000

Table 4. Fairness of Vegas-1, Vegas-2, Vegas-3, and Vegas1_3 (Min. RTT of connection 1 = 0.013211s).

Min. RTT of the longer connection	Vegas-1	Vegas-2	Vegas-3	Vegas1_3
0.231 s. (x2=110ms)	0.928	0.998	0.999	0.908
0.211 s. (x2=100ms)	0.898	1.000	1.000	0.892
0.191 s. (x2=90ms)	0.890	0.999	1.000	0.917
0.171 s. (x2=80ms)	0.829	1.000	1.000	0.953
0.151 s. (x2=70ms)	0.980	0.999	1.000	0.911
0.131 s. (x2=60ms)	0.973	1.000	1.000	0.887
0.111 s. (x2=50ms)	0.997	0.998	1.000	0.900
0.091 s. (x2=40ms)	1.000	1.000	1.000	0.900
0.071 s. (x2=30ms)	0.997	1.000	0.999	0.970
0.051 s. (x2=20ms)	0.984	0.999	0.999	0.925
0.031 s. (x2=10ms)	0.988	0.999	0.999	0.988
0.021 s. (x2=5ms)	0.994	0.999	1.000	0.980
0.013 s. (x2=1ms)	1.000	1.000	1.000	1.000

Table 5. Fairness of Vegas-1, Vegas-2, Vegas-3, and Vegas1_3 (Min. RTT of connection 1 = 0.013211s).

$\alpha = \beta = 3$	30% Vegas	50 % Vegas	70% Vegas
30 active flows	0.977 (234163/343386)	0.969 (293599/371919)	0.964 (323614/416524)
100 active flows	0.972 (114486/97053)	0.945 (107171/90024)	0.922 (99236/93750)
200 active flows	0.919 (67306/47123)	0.910 (56724/40596)	0.900 (49539/35516)

Table 6a. Simulation results when there is a mixture of Vegas-3 & TCP Reno, using RED-54-108 routers.

$\alpha = 1, \beta = 3$	30% Vegas	50 % Vegas	70% Vegas
30 active flows	0.960 (189427/347960)	0.844 (210509/454940)	0.759 (271019/888998)
100 active flows	0.964 (73244/101549)	0.920 (90460/106675)	0.883 (96956/116536)
200 active flows	0.949 (50734/48863)	0.916 (51865/45755)	0.891 (49217/41432)

Table 6b. Simulation results when there is a mixture of Vegas1_3 & TCP Reno, using RED-54-108 routers.

$\alpha = \beta = 3$	30% Vegas	50 % Vegas	70% Vegas
30 active flows	0.962 (354355/297238)	0.937 (301939/320963)	0.926 (320042/308864)
100 active flows	0.949 (110335/89808)	0.920 (110986/81527)	0.931 (103800/79588)
200 active flows	0895 (60808/41305)	0.888 (57531/36773)	0.901 (50991/35585)

Table 6c. Simulation results when there is a mixture of Vegas-3 & TCP Reno, using RED-5-15 routers.

$\alpha = 1, \beta = 3$	30% Vegas	50 % Vegas	70% Vegas
30 active flows	0.913 (237901/339379)	0.949 (312307/316247)	0.842 (290288/352538)
100 active flows	0.923 (113476/88255)	0.891 (101272/91257)	0.889 (97329/94748)
200 active flows	0.890 (58963/42945)	0.886 (54818/40014)	0.900 (51134/38425)

Table 6d. Simulation results when there is a mixture of Vegas1_3 & TCP Reno, using RED-5-15 routers.

Table 6. Simulation results for a mixture of Vegas1_3 & TCP Reno using RED routers.