11th IEEE International Conference on Computer Communications
and Networks (IC3N'02), Miami, Florida, October 2002. LA-UR 02-5522

On the Transient Behavior of TCP Vegas

Sarut Vanichpun
Department of Electrical & Computer Engineering
University of Maryland, College Park

Wu-chun Feng
Computer & Computational Sciences Division
Los Alamos National Laboratory

***Abstract*—Research has shown that TCP Vegas performs better than TCP Reno with respect to overall network utilization, stability, fairness, throughput, packet loss, and burstiness. In this paper, we analyze and improve the transient behavior of TCP Vegas, an important issue in today's large "bandwidth-delay product" networks. To quantify of our analysis, we introduce a new metric that captures the transient performance of TCP, namely, the (normailized) convergence time. We then consider the slow-start mechanism in TCP Vegas and show that with a properly configured γ parameter, the transient behavior of TCP Vegas improves with respect to convergence time.**

I. INTRODUCTION

Many Internet applications and application protocols, e.g., HTTP and FTP, use TCP as their transport layer. Consequently, researchers have focused much of their efforts in improving the performance of TCP resulting in two notable implementations of TCP — Reno [1] and Vegas [2].

To estimate the available bandwidth in the network, Reno uses packet loss as an indicator for congestion. This causes a periodic oscillation in the size of the congestion window and may not be appropriate for emerging Internet applications [3–5]. In contrast to Reno, Vegas performs better with respect to overall network utilization [2, 6], stability and fairness [7, 8], throughput and packet loss [2, 4, 6], and burstiness [3, 4]. However, its lack of widespread adoption can be attributed to its demonstrated incompatibility [7] with Reno.

Recently, Low [9] showed that Vegas achieves proportional fairness while Reno does not and that both have different utility and rate functions whence the incompatibility problem arises. However, we showed in [10] how to configure the congestion-control parameters, α and β, in Vegas to be compatible (and competitive) with Reno in the steady state. And in this paper, we extend that work and show that with a properly configured γ, Vegas can also be made compatible with Reno in the transient state.

With the current deployment of large "bandwidth-delay product" (BDP) networks, TCP research has focused on stability, throughput, and fairness — properties of the steady state. However, with these large BDP networks, the transient period of TCP can greatly affect overall performance, an issue of particular importance in distributed computational grids. For example, the time to converge to an optimal bandwidth value can take on the order of minutes in a large BDP network of 1 Gb/s $\times$ 100 ms = 100 Mb. Thus, if TCP's convergence mechanism is too passive, the network may be underutilized, and a given connection may not get its fair share of bandwidth.

In this paper, we introduce a metric to measure the transient behavior of TCP, namely, the (normalized) convergence time of TCP with respect to the uncongested BDP of the network. In particular, we focus on the transient behavior of TCP Vegas. With a careful analysis of the slow-start mechanism in Vegas, we show that the convergence time of TCP Vegas can be improved significantly with a properly configured γ.

The rest of the paper is organized as follows. Section II provides an overview of TCP Vegas. Section III discusses the convergence time of TCP protocols in general. In Section IV, we present an extensive analysis of the Vegas slow-start mechanism and suggest how to configure its γ parameter in order to achieve better performance. We confirm our analysis with simulation results in Section V. Lastly, we conclude and discuss future work in Section VI.

II. BACKGROUND ON TCP VEGAS

A. Congestion Avoidance

The bandwidth-estimation scheme in Vegas is proactive in that it tries to avoid rather than react to congestion (as in Reno). Vegas uses the difference in the expected and actual flow rates to estimate the available bandwidth in the network. When the network is uncongested, the actual flow rate is close to the expected flow rate. However, if the actual rate is much smaller than the expected rate, this indicates that buffer space in the network is filling up and that the network is approaching a congested state. This difference in flow rates can be calculated as $Diff = Expected - Actual$, where $Expected$ and $Actual$ are the expected and actual rates, respectively.

In its congestion-avoidance phase, Vegas uses two threshold values, α and β (whose default values are 1 and 3, respectively), to control the adjustment of the size of congestion window at the source host as follows. If d denotes the minimum-observed round-trip time (also known as $BaseRTT$), D denotes the actual round-trip time (RTT) of a packet, and W denotes the size of the congestion window, then $Expected = W/d$ and $Actual = W/D$. The estimated backlog of packets in the network queues can then be computed as

$$\Delta = (Expected - Actual) \times BaseRTT = W \times \frac{(D-d)}{D}. \quad (1)$$

For every RTT, the congestion-avoidance algorithm adjusts W

as follows:

$$W \leftarrow \begin{cases} W+1 & \text{if } \Delta < \alpha \\ W-1 & \text{if } \Delta > \beta \\ W & \text{otherwise } (\alpha \le \Delta \le \beta). \end{cases}$$

Conceptually, Vegas tries to keep at least α packets but no more than β packets queued in the network. Thus, when there is only one Vegas connection, W converges to a point that lies between $window + \alpha$ and $window + \beta$ where $window$ is the maximum window size that does not cause any queueing.

B. Slow Start

Like Reno, Vegas has a slow-start mechanism that allows a connection to quickly ramp up to the available bandwidth. However, unlike Reno, to ensure that the sending rate does not increase too fast during slow start, and hence, congest the network, Vegas doubles the size of its congestion window only *every other RTT*. In addition, every other RTT, Vegas calculates the difference in the flow rates ($Diff$) and Δ given in (1). When $\Delta > \gamma$ (whose default is 1), Vegas leaves its slow-start phase, drops its congestion window size by $1/8$ and enters its congestion-avoidance phase.[1]

III. Convergence Time

Most research in TCP considers only properties of the steady state, e.g., throughput, goodput, and fairness. However, with the deployment of large BDP networks, the transient properties of TCP are becoming increasingly important. In this section, we propose a metric to measure the convergence time (C_{time}) of TCP window protocols with respect to the uncongested BDP of the network. We then extend the analysis for the case of Vegas with an experimental study in Section V-B.

In order to define and understand the convergence time, we consider a simple case when only one connection tries to fill up an empty network of N links connecting the source and the destination. We denote the transmission rate of each of these N links (in packets/s) as $\mu_i, i = 1, \ldots, N$, and the total round-trip propagation delay of the connection (in seconds) as τ. Without loss of generality, we assume that $\mu_1 \le \mu_2 \le \ldots \le \mu_N$. Since μ_1 is the smallest transmission rate, link 1 behaves as a bottleneck link. The uncongested BDP of this network is then given by $\mu_1 d$ where

$$d = \tau + \frac{1+a}{\mu_1} + \ldots + \frac{1+a}{\mu_N} \qquad (2)$$

with a being the ACK size relative to the packet size.

Throughout our analysis, we assume that the TCP source is a fluid model and always has a packet to transmit and that the buffer sizes at routers are large enough so that packet loss is negligible. Moreover, we model the TCP-windowing mechanism with respect to a "round" [11], or equivalently, "window transmission." A round starts with the transmission of W packets (back-to-back) where W is the size of congestion window in that round. A round ends when the source receives the ACK of the first packet in that round, then the source starts transmitting a new packet of the next round.

Suppose that the total number of rounds in the transient period is k. For $i = 1, \ldots, k$, the time used by round i is given by $t_i = d + p_i$, i.e., the uncongested delay (d) plus the queueing delay in round i (p_i). Consequently, the transient period T is the sum over all $t_i, i = 1, 2, \ldots, k$ and is given by

$$T = \sum_{i=1}^{k} [d + p_i] = kd + \sum_{i=1}^{k} p_i. \qquad (3)$$

Because the transient behavior of TCP in networks with the same BDP are similar, k and p_i are constant for a given BDP and TCP algorithm. From (3), we can naturally define the transient convergence of a TCP algorithm as the (normalized) convergence time C_{time}:

$$C_{time} = \frac{T}{d}. \qquad (4)$$

C_{time} can be interpreted as the effective number of window transmissions (T/d) in the transient period since it indicates how many $BaseRTT$s are required to reach equilibrium. Thus, it represents how fast TCP behaves in the transient state. Moreover, C_{time} is a constant metric of a TCP algorithm with respect to BDP regardless of the number of links N used by the connection.

For Vegas, we can interpret C_{time} in a much simpler form using the fact that if the sending rate is smaller than the available rate of the link ($\frac{W}{d} \le \mu_1$), then $p_i = 0$. Recall that when a connection is in an empty network, Vegas has $p_i = 0$ at almost all i except the last few rounds that the congestion-window size exceeds the BDP of the network ($W > \mu_1 d$). Therefore, we approximate $\sum_{i=1}^{k} p_i \simeq 0$ and get

$$T \simeq kd \quad \text{and} \quad C_{time} \simeq k. \qquad (5)$$

IV. Analysis of Slow Start in Vegas

While the default parameter $\gamma = 1$ is appropriate for yesterday's small BDP networks, today's larger BDP networks allow the bandwidth allocated for each connection to be much larger, and hence, the equilibrium congestion-window size to be larger. In these larger BDP networks, Vegas with $\gamma = 1$ prematurely stops the exponentially-increasing, slow-start phase too early and enters the slower, congestion-avoidance phase (where only a linear increase every RTT is possible) until it reaches its equilibrium congestion-window size. As a result, Vegas has a very long transient period and the connection stays longer in the network than it should. (See, for example, Section V-A)

[1] However, in this case we *cannot* interpret the value of γ as the number of backlogged packets in the network during slow-start. (See Section IV for details.)

By properly selecting γ, we can control the duration of slow-start, and hence, the size of congestion window (W^*) that Vegas stops at upon leaving slow-start. In this section, we first derive the critical window size (W^*) for a given value of γ. Then, we propose two approaches for configuring γ to achieve better transient performance, i.e., better C_{time}.

A. Critical Window Size

Suppose that a Vegas connection uses N links to connect from the source to the destination in an empty network and that $\mu_i, i = 1, \ldots, N$ and τ are as defined in Section III. Without loss of generality, we again assume that $\mu_1 \leq \mu_2 \leq \ldots \leq \mu_N$. Furthermore, the assumptions on Vegas source and buffer sizes are similar to those given in Section III.

Recall that in slow-start, Vegas doubles its congestion window every other RTT. More precisely, in its "active" round, Vegas adds two packets in the sender queue each time an ACK of the previous round is received. Since μ_1 is the smallest transmission rate (and we assume that there is no congestion in ACK path), the spacing between each ACK of the previous round is $1/\mu_1$ seconds. When the last ACK of the previous round is received, the sender adds the last two packets of the active round in the sender queue. Since the transmission time at the sender queue for each packet is $\frac{1}{\mu_1}$, the last packet will see $\frac{W}{2}$ packets waiting ahead of it in the sender queue, where W is the size of congestion window in the active round. However, for the queues at other nodes along the connection, the last packet will see no packet in the queues because $1/\mu_1 \geq 1/\mu_2 \geq \ldots \geq 1/\mu_N$. Therefore, this last packet experiences the highest RTT of the round and its $BaseRTT$ and RTT are given by d in (2) and

$$D = d + \frac{W}{2\mu_1}, \tag{6}$$

respectively. For the analysis in the remainder of this section, we assume that a, the ACK size relative to the packet size, is small compared to 1, and we write only 1 for $1 + a$.[2]

By combining (6) and (1), TCP Vegas will stop its slow-start phase if

$$W \frac{W/2\mu_1}{W/2\mu_1 + d} > \gamma. \tag{7}$$

By solving (7) for W, the critical window size that TCP Vegas stops its slow-start phase is given by

$$W^*(\gamma) = \frac{\gamma + \sqrt{\gamma^2 + 8\mu_1 d\gamma}}{2}. \tag{8}$$

In an actual Vegas implementation, D is the average RTT rather than the actual RTT of a packet. Thus, D for the last packet is the average of the actual RTTs of all packets in the same round, i.e., $D = d + \frac{W}{4\mu_1}$, rather than $D = d + \frac{W}{2\mu_1}$, as given in (6). By using the average RTT, we have

$$W^*(\gamma) = \frac{\gamma + \sqrt{\gamma^2 + 16\mu_1 d\gamma}}{2}. \tag{9}$$

B. γ Selection

From (9), for any critical window size W^*, the "ideal" γ is given by

$$\gamma = \frac{W^{*2}}{4\mu_1 d + W^*}. \tag{10}$$

Although this ideal γ is specific to our simplified model, we use (10) as a suggestion for setting γ in general situations and verify by simulations that it is indeed useful.

From (10), we need to know μ_1 and W^* in order to set γ. In a realistic situation, μ_1 and W^* should be replaced by the available rate (λ) and the target window size ($\hat{W}$) of Vegas at the steady state. The reason for the above interpretation is that we want to configure γ so that the slow-start phase drives the congestion-window size to nearly $\hat{W}$ before changing over to the slow linear increase/decrease congestion-avoidance phase. At steady state, $\alpha \leq \Delta \leq \beta$, thus from (1), we have

$$\alpha \leq W - \lambda d \leq \beta \tag{11}$$

where $\lambda = \frac{W}{D}$ is the rate of TCP at steady state. From (11), we can *conservatively* set the target window as a function of λ by

$$\hat{W} = \lambda d + \alpha \qquad \text{, i.e., } \Delta = \alpha. \tag{12}$$

Now, let $W_i, i = 1, 2, \ldots, m$, be the size of the congestion window in round i during the slow-start phase where m is the last round of the slow-start phase. Because Vegas doubles its window size every other RTT, $W_i = W_{i+1}$ for $i = 1, 3, \ldots, m-1$, and these values can be determined based upon the knowledge of W_1. After Vegas exits slow start, its congestion-window size decreases to $\frac{7}{8}W_m$. From the analysis in Section IV-A, if we want Vegas to stop its slow start at W_m, then we should find γ such that

$$W_{m-2} < W^*(\gamma) \leq W_m. \tag{13}$$

From the above equations, we propose two approaches for setting γ. The first approach may still suffer from a large transient period T (although not nearly as large as when the default γ is used), but it is simpler to calculate and implement. The second approach reduces the transient period T, and hence, the convergence time C_{time} even further than the first approach.

1) Approach 1: Basic: If $\hat{W}$ lies between $[\frac{7}{8}W_i, \frac{7}{8}W_{i+2})$, we would like to stop the slow-start phase at $W_m = W_i$ and allow Vegas to linearly increase its window over the interval $[\frac{7}{8}W_i, \frac{7}{8}W_{i+2})$. Let $\bar{W}_i = \frac{W_{i-2}+W_i}{2}$ be the average window size in the interval $(W_{i-2}, W_i]$. By setting $W^* = \bar{W}_i$, we have

[2] Over our production network, our network monitoring showed that the ACK size was an order of magnitude smaller than the average packet size.

from (13) that Vegas stops its slow-start at W_i. From (10), we propose Approach 1 for setting γ, denoted by γ_1, as

$$\gamma_1 = \max(\lfloor \frac{\bar{W}_i^2}{4\lambda d + \bar{W}_i} \rfloor, 1) \quad \text{if } \hat{W} \in [\tfrac{7}{8}W_i, \tfrac{7}{8}W_{i+2}) \quad (14)$$

where $\lfloor x \rfloor$ denotes the floor of x and 1 is a lower bound for the value of γ. By choosing $\gamma = \gamma_1$, we allow the possibility of overshoot above the chosen $\hat{W}$ but ensure that only linear increase is used in congestion-avoidance mode to reach the steady state. As a result, when the actual target window size $\hat{W}$ is closer to the other end of the range, i.e., near $\frac{7}{8}W_{i+2}$, Vegas may still have a relatively large transient period due to the slow linear increase. The result of configuring $\gamma = \gamma_1$ is shown in Section V-D.

2) Approach 2: Enhanced: To get faster transient response, we allow both overshoot and linear decrease in the transient state. In this approach, we stop the slow-start phase at $W_m = W_i$ if $\hat{W} \in [\frac{7}{8}W_i, \frac{7}{8}\frac{W_i+W_{i+2}}{2})$ and at $W_m = W_{i+2}$ if $\hat{W} \in [\frac{7}{8}\frac{W_i+W_{i+2}}{2}, \frac{7}{8}W_{i+2})$. Vegas then performs linear increase if $W_m = W_i$ and linear decrease if $W_m = W_{i+2}$. By setting $W^* = \bar{W}_i$ and $W^* = \bar{W}_{i+2}$ for the first and second cases, respectively, we arrive at Approach 2 for setting γ:

$$\gamma_2 = \begin{cases} \max(\lfloor \frac{\bar{W}_i^2}{4\lambda d + \bar{W}_i} \rfloor, 1) & \text{if } \hat{W} \in [\frac{7}{8}W_i, \frac{7}{8}\frac{W_i+W_{i+2}}{2}) \\ \max(\lfloor \frac{\bar{W}_{i+2}^2}{4\lambda d + \bar{W}_{i+2}} \rfloor, 1) & \text{if } \hat{W} \in [\frac{7}{8}\frac{W_i+W_{i+2}}{2}, \frac{7}{8}W_{i+2}). \end{cases} \quad (15)$$

As we mentioned above, we need to know the available rate of the connection which is generally not available from the network. However, many "available bandwidth" probing methods have been proposed. For example, Allman and Paxson [12] reports available bandwidth probing techniques for initializing $ssthresh$ of Reno. In addition, the use of the next-generation active queue-management (AQM) schemes where the information can be transmitted back to the source by ECN [13], can provide this piece of information for setting γ.

V. Experiment

To verify our analysis, we run simulations of TCP Vegas using the discrete-event simulator *ns*, version 2.1b8a [14]. For all simulations, each connection generates FTP traffic with the packet size and ACK size fixed at 1 KB and 40 bytes, respectively.

A. Vegas in ns

In the slow-start phase for Vegas, the window size is updated as $1, 1, 3, 3, 6, 6, 12, 12, \ldots$, or equivalently,

$$W_i = \begin{cases} 1 & \text{for } i = 1, 2 \\ 3 \cdot 2^{\lfloor \frac{i-3}{2} \rfloor} & \text{for } i = 3, 4, \ldots, \end{cases} \quad (16)$$

where W_i denotes the size of congestion window in round i.

With the default value of γ, i.e., $\gamma = 1$, Figure 1 shows the transient behavior of Vegas for an the empty network with $\mu = 10,000$ packets/second and $\tau = 20$ ms. In this case, the transient period is $T = 3.54$ seconds, and the convergence time $C_{time} \approx 176$ which is very long since approximately 176 RTTs are needed to reach the steady state. This slow convergence behavior is due to the fact that TCP Vegas stops its slow-start phase earlier than it should and changes to the slower linear increase of the congestion-avoidance phase.

TABLE I

W^* WITH DIFFERENT γ IN TWO-LINK NETWORK WITH $\mu_1 = 5,000$ PACKETS/S, $\mu_2 = 50,000$ PACKETS/S AND $\tau = 40$ MS

d	BDP	γ	W^*	$W_{m-2}(ns)$	$W_m(ns)$
40.2288	201.1	1	28.87	24	48
40.2288	201.1	2	41.12	24	48
40.2288	201.1	5	65.98	48	96
40.2288	201.1	8	84.33	48	96
40.2288	201.1	12	104.44	96	192

B. Convergence Time

Figure 2 shows C_{time} of a Vegas connection with $\gamma = 1$ with no competing connections. The notation "1link" indicates the results in a one-link network while "mlink" represents the results in a multiple-link network.[3] Clearly, the values of C_{time} are independent of the number of links and are constant for each value of BDP. Hence, C_{time} is a performance measure of TCP algorithms in the transient state. Again, we note that by fixing $\gamma = 1$, Vegas requires an exorbitant amount of time to converge to the available bandwidth as the BDP increases. Thus, the default configuration of Vegas is not appropriate for networks with a high BDP.

C. Critical Window Size

In order to verify (9), we consider the case of one Vegas connection in a network with two links having transmission rates μ_1 and μ_2 packets/s, respectively and vary the values of γ. Our analysis is "correct" if W^* satisfies (13), i.e., it lies between W_{m-2} and W_m where W_m is the congestion window size in the round that Vegas stops its slow-start phase. The results in Table I follows the analysis for all values of γ.

D. γ Selection

Figure 3 shows that with the proper configuration of γ, the convergence time of Vegas improves dramatically, particularly for large BDP networks. By choosing $\gamma = \gamma_1$ (i.e., ct1 in Figure 3), we see that Vegas still suffers from a relatively long convergence time because the equilibrium window size is quite

[3] We ran simulations for many different number of links and get virtually the same results in each case.. Here, we represent a multiple-link network using a 2-link network.

far from the value of the window size that Vegas stops its slow-start phase; this is especially true in larger BDP networks. By setting $\gamma = \gamma_2$ (i.e., ct2 in Figure 3), Vegas sustains or improves the convergence times over all the networks.

Now, we show an example of the improved transient behavior of Vegas with a properly configured γ. Consider the situation when four connections compete in a bottleneck link with $\mu_1 = 10,000$ packets/s. Each connection connects to the bottleneck link using a link with $\mu_2 = 50,000$ packets/s and has total propagation delay $\tau = 20$ ms. The first two connections start at time 0 seconds with $\gamma = \gamma_1 = 5$ for $\hat{W} = 101.6$ and the last two connections start at time 2 seconds with $\gamma = \gamma_1 = 1$ for $\hat{W} = 51.3$.[4] Figure 4 shows the results of this experiment. When there are two connections, both connections stop slow start at the window size of 96 and converge near $\hat{W}$ thus having small convergence times. However, when there are four connections, the new connections stop their slow-start phases at a window size of 24 (shortly after time $t = 2$ s) while we expect them to stop at the window size of 48. As a result, all connections take a long time before converging to the steady state.

The above problem can be explained as follows: Since the network is already fully utilized, the $BaseRTT$ of the new connection is larger than that of the existing connection starting in an empty network. Thus, Δ in slow-start mode get large faster and the connection stops the slow-start at the smaller size of congestion window than W^* as analyzed in (9). Hence, our prediction of γ tends to be conservative as needed in the realistic situation (i.e., preventing a packet loss). This also suggests that to further improve the transient period of Vegas, we may need to modify its congestion-avoidance algorithm.

VI. Conclusion

Prior research has focused on the steady-state behavior of TCP algorithms. In contrast, this paper addresses the transient behavior of TCP algorithms and introduces a new metric — the (normalized) convergence time C_{time} — to measure the transient performance of TCP. In particular, we show how inappropriate the default configuration of Vegas ($\gamma = 1$) is in high BDP networks and propose two approaches to set γ appropriately so that Vegas can achieve faster transient response without causing any packet loss.

To set γ, however, Vegas needs to estimate the available bandwidth of the network at the steady state. As discussed at the end of Section IV-B, our future work will incorporate bandwidth-estimation schemes and AQM routers along with a form of ECN to provide Vegas with the necessary information to set γ. Further, we will extend our analysis to the case when many connections are present in the network and examine how to improve the congestion-avoidance phase to achieve not only better convergence time but also better throughput and fairness between connections.

[4] The values of $\hat{W}$ and γ are chosen so that each connection gets a fair share of bandwidth when there are two and four connections, respectively.

References

[1] V. Jacobson, "Modified TCP Congestion Avoidance Algorithm," Technical Report, April 1990.

[2] L. Brakmo and L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet," *IEEE Journal on Selected Areas in Communication*, vol. 13, no. 8, pp. 1465–1480, October 1995.

[3] A. Veres and M. Boda, "The Chaotic Nature of TCP Congestion Control," in *Proc. of IEEE INFOCOM 2000*, March 2000.

[4] W. Feng and P. Tinnakornsrisuphap, "The Failure of TCP in High-Performance Computational Grids," in *Proc. of SC2000: High-Performance Networking and Computing Conf.*, November 2000.

[5] Y. R. Yang, M. S. Kim, and S. S. Lam, "Transient Behaviors of TCP-friendly Congestion Control Protocols," ftp://ftp.cs.utexas.edu/pub/lam/transient_tech.ps.gz, July 2000.

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

[7] J. Mo, R. J. La, V. Anantharam, and J. Walrand, "Analysis and Comparison of TCP Reno and Vegas," in *Proc. of IEEE INFOCOM 1999*, March 1999, pp. 1556–1563.

[8] G. Hasegawa, M. Murata, and H. Miyahara, "Fairness and Stability of Congestion Control Mechanisms of TCP," in *Proc. of IEEE INFOCOM 1999*, March 1999, pp. 1329–1336.

[9] S. Low, "A Duality Model of TCP and Queue Management Algorithms," in *Proc. of ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management*, September 2000.

[10] E. Weigle and W. Feng, "A Case for TCP Vegas in High-Performance Computational Grids," in *IEEE International Symposium on High Performance Distributed Computing*, August 2001.

[11] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP Throughput: A Simple Model and Its Empirical Validation," *IEEE/ACM Transactions on Networking*, vol. 8, no. 2, pp. 133–145, 2000.

[12] M. Allman and V. Paxson, "On Estimating End-to-End Network Path Properties," in *Proc. of ACM SIGCOMM 1999*, August 1999, pp. 263–274.

[13] K. Ramakrishnan and S. Floyd, "A Proposal to Add Explicit Congestion Notification (ECN) to IP," RFC 2481, January 1999.

[14] S. McCanne and S. Floyd, "ns Network Simulator, version 2.1b8a," http://www.isi.edu/nsnam/ns.

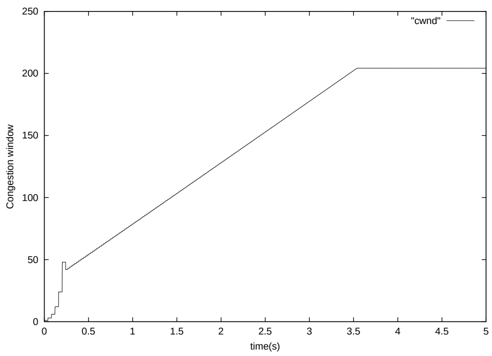

Fig. 1. Example of transient behavior of Vegas with $\gamma = 1$

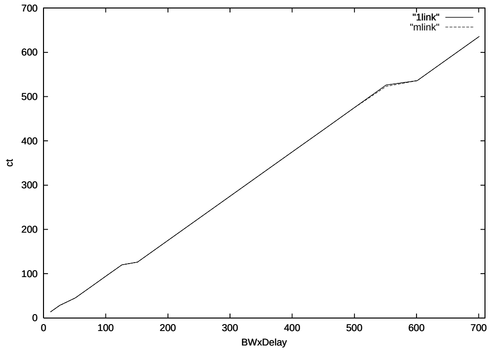

Fig. 2. C_{time} of TCP Vegas with $\gamma = 1$

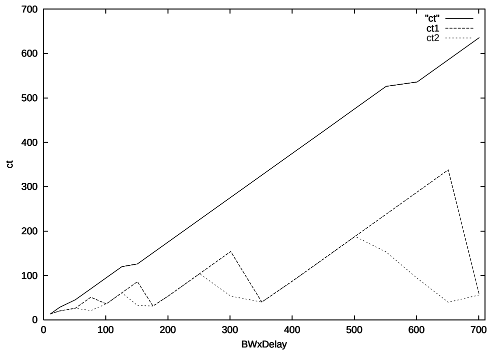

Fig. 3. Convergence time of Vegas. (ct, $ct1$, and $ct2$ denote C_{time} of Vegas with $\gamma = 1$, $\gamma = \gamma_1$, and $\gamma = \gamma_2$, respectively.)

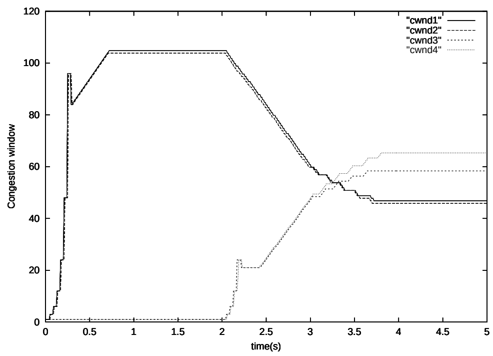

Fig. 4. Congestion windows of 4 Vegas connections in the bottleneck network. Two connections start at 0 s and another two connections start at 2 s