Modeling a Mixed TCP Vegas and TCP Reno Scenario

Andrea De Vendictis and Andrea Baiocchi

INFOCOM Dept. - University of Roma *La Sapienza*
{devendictis,baiocchi}@infocom.uniroma1.it

Abstract. In this paper we describe and validate the analytic model of a mixed TCP Reno and TCP Vegas network scenario. There is experimental evidence that TCP Vegas overcomes the widespread TCP version, called TCP Reno, in a number of network environments. The incompatibility between TCP Vegas and TCP Reno in heterogeneous network scenarios has been also verified by means of several simulations. The model presented in this work allows to quantitatively evaluate this incompatibility, by computing the average throughput of a TCP Vegas source in presence of a concurrent TCP Reno source. This model can help us to better understand the reasons of the vulnerability of TCP Vegas in competing with TCP Reno sources.

1 Introduction

In the last years TCP congestion control has received great interest from the networking research community. A number of analytic and experimental studies have pointed out the shortcomings of TCP and they conceived changes able to cope with some TCP limitations. The recent standardized modifications are a result of these efforts (for an updated report see [1]). However, most of these modifications concerned improvements in avoiding unnecessary timeouts and fast retransmit caused by packet reordering, isolated packet loss and multiple packet loss due to temporary network congestion. Hence, the changes are intended in order to optimize the mechanisms that regulate the response to loss detection. Conversely, the mechanisms that avoid protracted network congestion (slow start and congestion avoidance) and define how TCP sources share the available bandwidth among them are quite unchanged (at least as standards).

TCP Vegas, proposed in [2], represents a valid alternative to the congestion control performed by the currently standard and most widespread version of TCP, called TCP Reno. Although it introduces new techniques into all the main mechanisms of TCP, it is fully compatible with all the standard versions of TCP, because the changes only concern the TCP sending side.

TCP Reno congestion control reacts to the network congestion only after loss detection, thus when the network congestion has already arisen. This avoids congestion collapse, since the transmission rate is reduced as soon as a packet loss occurs; however this generates an intrinsic instability of the congestion control, whose evidence is the permanent oscillation of the source transmission rate.

E. Gregori et al. (Eds.): NETWORKING 2002, LNCS 2345, pp. 612–623, 2002.

The key idea of TCP Vegas is to prevent the packet loss (and the network congestion) by adapting the transmission rate to the value of the round trip delay experienced by the transmitted packets.

Several simulation works have verified that TCP Vegas is better than TCP Reno in terms of throughput (between 37% and 71% better than Reno), fairness, stability, packet loss probability, end-to-end delay and ability in avoiding network congestion in a very large number of network environments [2][3][4][5].

However, some works [3][4][6] also pointed out by means of simulations and experimental trials the extreme vulnerability of TCP Vegas in competing with TCP Reno sources to take the available bandwidth. The problem is that TCP Reno is intrinsically much more aggressive than TCP Vegas, because it reduces its transmission rate only after packet loss detection.

This represents the main reason why TCP Vegas cannot be widely proposed to the users as a reliable transport protocol.

Given that TCP Vegas basic approach is a sound one, there is a need for a thorough understanding of the fine tuning of its parameters and possibly for some modifications of the basic congestion control algorithm. To this end it is useful to have an accurate analytic tool for the evaluation of TCP Vegas in a mixed TCP environment.

This paper describes an analytic model to evaluate the average throughput of a TCP Vegas source interacting with a TCP Reno source in a mixed scenario.

Since the works [3][7] already modeled the competition of TCP Vegas and TCP Reno in a network environment with small bandwidth-delay product (less than the bottleneck link buffer size), we will only analyze the case of a network with large bandwidth-delay product (larger than the buffer size). This is even more interesting as the small delay-bandwidth product case, since optical networking and high speed processing devices promise very high network capacity also in the wide area network environment.

The aim is to give a quantitative measure of the vulnerability of TCP Vegas in presence of a source implementing TCP Reno and to gain insight as to what might be acted upon to reverse this outcome.

This study can represent a starting point to find mechanisms that make TCP Vegas competitive in a heterogeneous network scenario: given the distributed nature of the Internet, this is a key element to stimulate the use of TCP Vegas as a reliable transport protocol and to improve the TCP congestion control.

The rest of the paper is structured as follows. In Section 2 the TCP Reno and TCP Vegas congestion control is described. In Section 3 we present the analytic model. In Section 4 the validation of the model is shown. Finally, in Section 5 the conclusions and hints to further work.

2 TCP Vegas and TCP Reno Congestion Control

TCP Reno and TCP Vegas adopt an end-to-end closed-loop adaptive window congestion control. It is based on five fundamental mechanisms: slow start, congestion avoidance, retransmission time-out, fast retransmit and fast recovery.

TCP Reno and TCP Vegas use slow start at the beginning of the connection and whenever a packet loss is detected via timeout.

When the congestion window reaches a threshold value (called *slow start threshold*), TCP Reno and TCP Vegas enter congestion avoidance.

The maximum limit of the congestion window is advertised by the receiver to the sender during the connection.

Both the protocols can detect packet losses by means of two mechanisms: when the timeout (set when the packet is sent) expires, they reduce their congestion window to one packet size[1], then they start again in slow start[2]. Otherwise, if three duplicated acknowledgments arrive back to the sender before the timeout expiration, the protocols perform fast retransmit and fast recovery.

TCP Vegas differs from TCP Reno by the way slow start, congestion avoidance and fast retransmit are implemented.

During slow start, TCP Reno congestion window W_R increases by one packet for every incoming acknowledgment. So the congestion window has an exponential growth. During congestion avoidance TCP Reno opens its congestion window W_R linearly, because for each incoming acknowledgment it updates the congestion window by incrementing it by $1/W_R$.

In fast recovery TCP Reno halves the current congestion window and goes in congestion avoidance, without performing slow start.

The TCP Vegas congestion control is based on two parameters representing respectively the *expected* and the *actual* rate, calculated as follows:

$$Expected = \frac{W_V}{BaseRTT} \quad Actual = \frac{FlightSize}{RTT}$$

where W_V is the value of the current congestion window, $BaseRTT$ is the minimum round trip time experienced by the connection, $FlightSize$ is the number of packets currently not acknowledged yet, RTT is the round trip time experienced by the considered packet.

For each incoming acknowledgment, TCP Vegas computes the normalized difference $Diff$ between $Expected$ and $Actual$:

$$Diff = (Expected - Actual) \cdot BaseRTT \tag{1}$$

During congestion avoidance, TCP Vegas compares $Diff$ with two thresholds α and β[3]. TCP Vegas updates its congestion window W_V as follows:

$$W_V = \begin{cases} W_V + \frac{1}{W_V} & if \quad Diff < \alpha \\ W_V - \frac{1}{W_V} & if \quad Diff > \beta \\ W_V & otherwise \end{cases} \tag{2}$$

The slow start mechanism is based on the same concepts of congestion avoidance: TCP Vegas computes $Diff$ and compares it with a unique threshold γ[4]; as long as $Diff$ is less than γ or W_V is less than the slow start threshold, TCP

[1] From now on, we measure the window size in number of packets

[2] Actually, the initial slow start congestion window value depends on the implementation for both TCP Reno and TCP Vegas.

[3] Suggested values are $\alpha = 1$ and $\beta = 3$

[4] Suggested value is $\gamma = 1$.

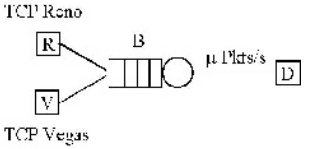

Fig. 1. Network model.

Vegas increases its congestion window by one packet every other round trip delay. Then, TCP Vegas performs congestion avoidance.

After fast retransmit, TCP Vegas reduces the window threshold to the half of the current congestion window and its window by a factor 3/4.

We finally observe that the retransmission mechanisms of TCP Vegas are enhanced with respect to TCP Reno because of the use of a fine-grained clock.

3 The Model

We are interested in analyzing the behavior of the TCP Reno and TCP Vegas mechanisms under the same conditions and with interacting sources exploiting the two protocols.

Therefore, we consider a network scenario with two isolated TCP sources sharing a common route. One source implements the TCP Reno variant, whereas the other the TCP Vegas variant. We assume the sources always have data to send and the transmitted packets have all the same length L.

Along the forward path there is a single bottleneck link with capacity μ packets/sec and a FIFO buffer of size B packets. The others links (including those crossed by the TCP acknowledgments) have a capacity such that they do not limit the source throughput.

The sources experience the same propagation delay T, because they have the same route and destination. For propagation delay T we mean here the time elapsing since the transmission of the packet from the source and the arrival of the corresponding acknowledgment, with the exclusion of the waiting time in the bottleneck buffer. The figure 1 depicts the reference network scenario.

We assume the following hypotheses:

1. The receiver does not pose any limitation to the value of the sender congestion window.
2. The bandwidth-delay product is larger than the buffer size ($\mu T > B$).
3. The sources detect a packet loss via duplicate acknowledgment by using the fast retransmit and fast recovery algorithms. Thus, after observing a packet loss the sources perform the congestion avoidance algorithm.
4. The buffer is empty at the beginning of the congestion avoidance.
5. The sources experience packet loss almost simultaneously; therefore they begin the congestion avoidance together.

These hypotheses lead to a cyclical model for the steady-state behavior of the sources. In figure 2 the periodic evolution of the congestion windows is shown

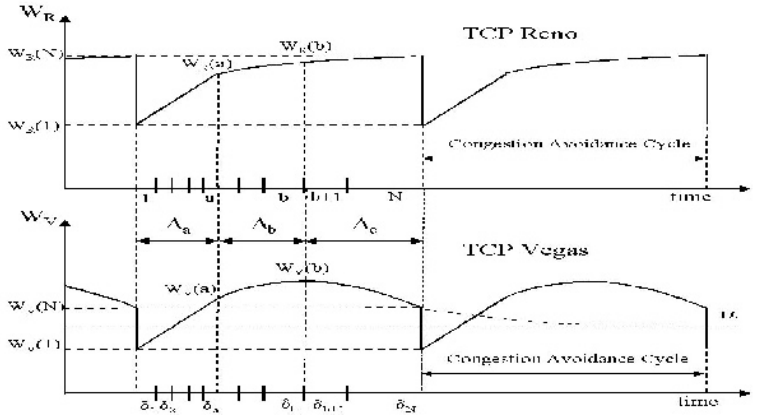

Fig. 2. Congestion Window Evolution of TCP Reno and TCP Vegas.

respectively for TCP Reno (upper plot) and TCP Vegas (lower plot).

Each cycle, called from now on congestion avoidance cycle, begins after a packet loss is detected. During each cycle the congestion avoidance algorithm regulates the window evolution.

By applying the approach already adopted in [9], we consider a congestion avoidance cycle divided in N mini-cycles of duration $\{\delta(i), i = 1, 2, 3, ..., N\}$. The mini-cycle is the time interval during which the same window is maintained. The value of the window during the i-th mini-cycle is denoted by $W(i)$. Formally, the mini-cycles are defined as follows: $\delta(1)$ is T, $\{\delta(i), i = 2, ..., N\}$ is the time elapsing since reception of the acknowledgment of the last packet sent in the window $W(i-1)$ and the arrival of the acknowledgment of the last packet sent in the window $W(i)$.

As for TCP Vegas, we assume in our analysis it has a unique threshold, i.e. $\alpha = \beta$. This simplification does not strongly change the nature of TCP Vegas, because it only eliminates the stable interval without modifying the principles of its mechanisms. Moreover, we can assume that $FlightSize \cong W_V$ and $BaseRTT \cong T$. Therefore, from (2) the congestion window W_V is updated for every incoming acknowledgment as follows:

$$W_V = \begin{cases} W_V + \frac{1}{W_V} & if \quad \left(\frac{W_V}{T} - \frac{W_V}{RTT}\right) \cdot T \leq \alpha \\ W_V - \frac{1}{W_V} & if \quad \left(\frac{W_V}{T} - \frac{W_V}{RTT}\right) \cdot T > \alpha \end{cases} \tag{3}$$

In order to derive the throughput of the sources we must calculate the number of packets sent during a single congestion avoidance cycle and its duration.

First, we calculate the window size of the sources at the end of each cycle, because they affect all the window dynamics.

Let $\{W_V(i), i = 1, ..., N\}$ and $\{W_R(i), i = 1, ..., N\}$ be the size of the i-th congestion window respectively for TCP Vegas and TCP Reno.

To find the final value $W_V(N)$ of the TCP Vegas congestion window, we must consider the behavior of TCP Vegas congestion control.

Let μ_V be the available bandwidth for TCP Vegas:

$$\mu_V = \frac{W_V}{W_R + W_V}\mu \tag{4}$$

If q is the average size of the queue experienced by the packet corresponding to the incoming acknowledgment:

$$W_V + W_R = \mu T + q \tag{5}$$

From (3), TCP Vegas tries to reach an equilibrium in which its congestion window W_V is such that:

$$\left(\frac{W_V}{T} - \frac{W_V}{RTT}\right) \cdot T = \alpha \tag{6}$$

Because W_V/RTT can be interpreted as the available bandwidth μ_V, from (4), (5) and (6), the equilibrium value for the TCP Vegas congestion window is:

$$W_V = \frac{\alpha\,(\mu T + q)}{q} \tag{7}$$

Since at the end of the congestion avoidance phase we can assume $q = B$:

$$W_V(N) = \frac{\alpha\,(\mu T + B)}{B}$$

Since the total number of packets that can be accommodated in the network is $\mu T + B$, we observe a packet loss when $W_V(N) + W_R(N) = \mu T + B + 1$.

Hence, the congestion window of the Reno source at the end of the congestion avoidance cycle is:

$$W_R(N) = (\mu T + B)\left(\frac{B - \alpha}{B}\right) + 1$$

Because in the TCP Reno variant the congestion window is halved after detecting a packet loss via triple duplicate acknowledgments, we have for its initial congestion window size $W_R(1)$:

$$W_R(1) = \frac{W_R(N)}{2} \tag{8}$$

Instead, in TCP Vegas the congestion window $W_V(1)$ is generally set to 3/4 of the final congestion window $W_V(N)$. However, since we want to analyze the impact on the TCP Vegas performance of the amount of the window decreasing, we can generically assume:

$$W_V(1) = \lambda \cdot W_V(N) \tag{9}$$

with λ a positive constant less than 1.

Because during congestion avoidance TCP Reno congestion window $W_R(i)$ grows by 1 packet every mini-cycle:

$$W_R(i) = W_R(1) + i - 1 \quad for \quad i = 1, ..., N \tag{10}$$

Thus, the total number P_{Reno} of packets transmitted from the TCP Reno source during a single congestion avoidance cycle is:

$$P_{Reno} = \sum_{i=1}^{N} W_R(i) = N \cdot W_R(1) + \frac{N(N-1)}{2}$$

To calculate the number of packets transmitted by TCP Vegas during a single congestion avoidance cycle and its duration, we can distinguish three phases:

1. From mini-cycle 1 to a: as long as $W_V + W_R \leq \mu T$ we can assume there is no packets in the bottleneck buffer, so $\{\delta(i) = T, i = 1, ..., a\}$. TCP Vegas increases its congestion window by 1 packet every T, because $RTT \cong T$.
2. From mini-cycle $(a + 1)$ to b: the duration $\{\delta(i), i = a + 1, ..., b\}$ of the mini-cycles increases because of the queuing delay experienced by the transmitted packets; however TCP Vegas still enlarges its window by 1 packet every $\delta(i)$, because it has not reached the equilibrium value in (7) yet.
3. From mini-cycle $(b + 1)$ to N: TCP Vegas reduces its congestion window to preserve the equilibrium in (7) while the queuing size increases because of the TCP Reno source.

In the first phase the congestion windows increase for each mini-cycle by one packet and the window growth is identical for the two sources; then, the number a of mini-cycles of duration T is:

$$a = (W_V(a) - W_V(1) + 1) = (W_R(a) - W_R(1) + 1) \tag{11}$$

When the link is saturated, the sum of the windows of the two sources is:

$$W_V(a) + W_R(a) = \mu T \tag{12}$$

From (11) and (12):

$$W_V(a) = \frac{1}{2}(\mu T + W_V(1) - W_R(1))$$
$$W_R(a) = \frac{1}{2}(\mu T + W_R(1) - W_V(1))$$
$$a = \frac{1}{2}(\mu T - W_R(1) - W_V(1)) + 1$$

The number of packets sent by TCP Vegas during the first phase and its duration Δ_a are respectively:

$$A_V = \sum_{i=1}^{a} W_V(i) = a \cdot W_V(1) + \frac{a(a-1)}{2} \quad ; \qquad \Delta_a = a \cdot T$$

After the first phase the available bandwidth μ of the bottleneck link is divided between the two sources proportionally to the their respective windows.

Thus, the TCP Vegas available bandwidth $\mu_V(i)$ at the i-th mini-cycle is:

$$\mu_V(i) = \frac{W_V(i)}{W_V(i) + W_R(i)} \cdot \mu \quad for \quad i = a+1, ..., N \tag{13}$$

The duration of each mini-cycle is:

$$\delta(i) = \frac{W_V(i)}{\mu_V(i)} \qquad for \quad i = a+1, ..., N \tag{14}$$

From (13) and (14):

$$\delta(i) = \frac{W_V(i) + W_R(i)}{\mu} \qquad for \quad i = a+1, ..., N \tag{15}$$

where the sum of the windows along the second phase is obtained by considering that the average queuing size increases by two packets every mini-cycle:

$$W_V(i) + W_R(i) = \mu T + 2(i-a) \qquad for \quad i = a+1, ..., b$$

From (15), the duration Δ_b of the second phase is:

$$\Delta_b = \sum_{i=a+1}^{b} \delta(i) = \left(T + \frac{b-a+1}{\mu}\right)(b-a)$$

The number of packets sent by TCP Vegas during Δ_b is:

$$B_V = (b-a)W_V(a) + \frac{(a+1+b)(b-a)}{2}$$

From (7), TCP Vegas begins to reduce its congestion window when the queuing size q is $2(b-a)$, with b satisfying the following equation:

$$W_V(b) = \frac{\alpha\left(\mu T + 2(b-a)\right)}{2(b-a)} \tag{16}$$

Since $W_V(b) = W_V(a) + b - a$, we calculate b by solving (16):

$$b = a + \frac{-\left(W_V(a) - \alpha\right) + \sqrt{\left(W_V(a) - \alpha\right)^2 + 2\alpha\mu T}}{2}$$

The window size of TCP Reno is $W_R(b) = W_R(a) + b - a$. In the third phase, the queuing size in each mini-cycle can increase at most by 1 packet. Thus, the congestion window of TCP Vegas can follow the equilibrium value given in (7). Thus, by solving (7) with $q = W_V(i) + W_R(b) + i - b - \mu T$, $W_V(i)$ is:

$$W_V(i) = \frac{-C(i) + \sqrt{C(i)^2 + 4\alpha\left[W_R(b) + i - b\right)]}}{2} \qquad for \quad i = b+1, ..., N \tag{17}$$

where $C(i) = W_R(b) + i - b - \alpha - \mu T$.

From (15), the duration Δ_c of the third phase is:

$$\Delta_c = \sum_{i=b+1}^{N} \delta(i) = \sum_{i=b+1}^{N} \frac{W_R(i) + W_V(i)}{\mu}$$

with $W_R(i)$ and $W_V(i)$ respectively calculated according to (10) and (17).

The number of packets sent by TCP Vegas during the third phase is:

$$C_V = \sum_{i=b+1}^{N} W_V(i)$$

Therefore, the average throughput Λ_{Vegas} of TCP Vegas is the ratio of the total number P_{Vegas} of transmitted packets to the time Δ:

$$\Lambda_{Vegas} = \frac{P_{Vegas}}{\Delta} = \frac{A_V + B_V + C_V}{\Delta_a + \Delta_b + \Delta_c}$$

The average throughput Λ_{Reno} of TCP Reno is:

$$\Lambda_{Reno} = \frac{P_{Reno}}{\Delta} = \frac{P_{Reno}}{\Delta_a + \Delta_b + \Delta_c}$$

The above analysis is valid if the buffer size is sufficient to allow TCP Vegas to reach the equilibrium status expressed by (7). If this is not the case, we have again the first and the second phases shown if figure 2, whereas the third phase is absent. When the buffer size is less than a threshold B_{th}, TCP Vegas behaves like TCP Reno as already argued in [10], and the average throughput of the two sources only depends on how much TCP Vegas reduces its congestion window after detecting loss, i.e. on the amount of λ. For instance, if $\lambda = 1/2$ the two sources are perfectly equivalent and they experiment the same average throughput. Otherwise, if $\lambda > 1/2$ TCP Vegas pushes more packets than TCP Reno into the pipe. The technique used to calculate the average throughput of the two sources in this case is similar to that adopted above and is shown in Appendix A. The buffer threshold B_{th} is calculated by imposing that the final TCP Vegas congestion window $W_V(N)$ is equal to the congestion window at the equilibrium:

$$W_V(N) = \frac{(\mu T + B_{th} + 1)}{3 - 2\lambda} = \frac{\alpha(\mu T + B_{th})}{B_{th}}$$

Thus, the buffer threshold $B_{th} \cong \alpha(3 - 2\lambda)$.

4 Validation of the Model

In order to validate the analytic model, we carried out simulations under *ns* [11], a network simulator widely used in the networking research community.

The network topology used in the simulations reproduces that shown in figure 1. The two sources start at time 0 and the simulation lasts 600 seconds.

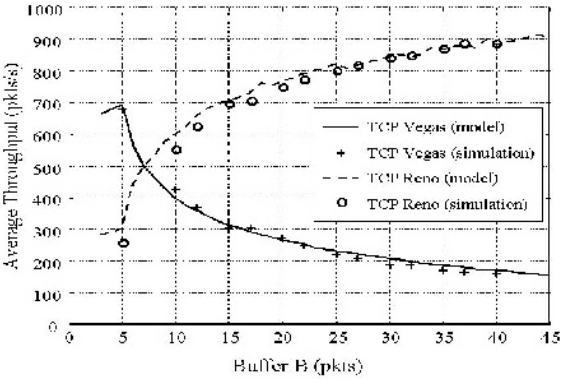

Fig. 3. Model Validation - Varying the buffer size B ($\mu = 1080 pkts/s$, $\alpha = \beta = 3$, $\lambda = 3/4$, $d = 500$ *bytes*)

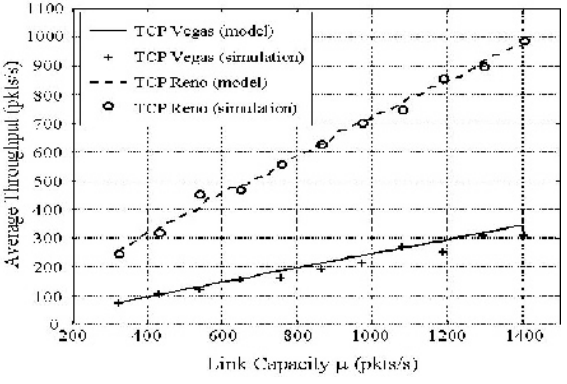

Fig. 4. Model Validation - Varying the link capacity μ (B=20, $\alpha = \beta = 3$, $\lambda = 3/4$, $L = 500$ *bytes*)

The source average throughput is calculated over the last 400 seconds, in order to capture the steady-state dynamics. In the simulations the sources exhibit a steady-state periodic behavior, as assumed in our model.

Given the particular *ns* implementation of TCP Vegas, we could not compare our analytic model with the simulations for buffer size between the threshold B_{th} and $[B_{th} + 3(3 - 2\lambda)]$. The reason of this is that the implementation is such that TCP Vegas congestion window oscillates between the equilibrium (7) and the equilibrium plus three packets. This phenomenon is negligible for larger buffers.

In figure 3 we show the model performance obtained by varying the buffer size B. The other parameters in these simulations are fixed to: $\mu = 1080\ pkts/s$, $\alpha = \beta = 3$, $\lambda = 3/4$, $L = 500$ *bytes*.

The model captures very well the behavior of the sources. In particular, the dynamics of TCP Vegas are faithfully reproduced with the maximum absolute percentage error of 12.2% (Buffer B=30pkts), whereas as for TCP Reno the maximum absolute percentage error is of 9.9% (Buffer B=5pkts).

As we expected, for small buffers TCP Vegas outperforms TCP Reno. In fact, under the buffer threshold (in this case the threshold B_{th} was 5) TCP Vegas be-

haves like TCP Reno, because it is not able to reach a stable status. Moreover, TCP Vegas reduces its congestion window after the fast retransmit by $\lambda = 3/4$, whereas TCP Reno halves it.Hence TCP Vegas goes better.

According to the model, the behavior of the sources changes quickly around the buffer threshold B_{th}: it is sufficient to change the buffer from 5 packets to 10 packets to change from a throughput ratio of $\frac{\Lambda_{Vegas}}{\Lambda_{Reno}} = 2.58$ to a throughput ratio of $\frac{\Lambda_{Vegas}}{\Lambda_{Reno}} = 0.76$. Thus, for large buffers TCP Vegas performance quickly deteriorate: its average throughput tends to go to zero.

In figure 4 we show the average throughput of the two sources obtained by varying the bottleneck link capacity μ. The other parameters are fixed in this case to: $B = 20\ pkts$, $\alpha = \beta = 3$, $\lambda = 3/4$, $L = 500\ bytes$.

Also in this case the model captures very well the behavior of the two sources.

According to the model the ratio of TCP Vegas throughput to TCP Reno throughput goes lightly reducing itself when the capacity μ increases. However the slope is too slow to be appreciated in the simulation results.

5 Conclusions

In this paper we presented an analytic model to compute the average throughput of two interactive sources, one implementing as transport protocol the TCP Vegas variant, the other implementing the TCP Reno variant. The aim of the model is to quantitatively establish the vulnerability of TCP Vegas with respect to a concurrent TCP Reno source.

The simulations we carried out show the large accuracy of the model in capturing TCP Vegas and TCP Reno dynamics.

Even if the model reproduces a very simple network scenario with only two sources, it can give interesting indications about TCP Vegas shortcomings.

As further work, our intention is to use the model as a base point to find possible mechanisms to improve TCP Vegas performance in scenarios where there are also TCP Reno sources, in order to stimulate the use of TCP Vegas in the future Internet.

Appendix A

When the buffer B is less than the threshold B_{th}, TCP Vegas and TCP Reno behave in the same way: they increase their congestion window by 1 packet for each mini-cycle until a packet loss occurs. Thus, they have respectively $W_V(N) = W_V(1) + N - 1$ and $W_R(N) = W_R(1) + N - 1$. Since:

$$W_V(1) = \lambda W_V(N); \quad W_R(1) = \frac{1}{2} W_R(N); \quad W_V(N) + W_R(N) = \mu T + B + 1$$

the number N of mini-cycles and the packets transmitted by the sources are:

$$N = \frac{(\mu T + B + 1)(1 - \lambda)}{3 - 2\lambda} + 1$$

$$P_{Reno} = \sum_{i=1}^{N} W_R(i) = \frac{3}{2} N(N-1); \quad P_{Vegas} = \sum_{i=1}^{N} W_V(i) = \frac{1+\lambda}{2(1-\lambda)} N(N-1)$$

We distinguish two phases to calculate the duration of the congestion avoidance cycle: from mini-cycle 1 to mini-cycle a we have $\delta(i) = T$, because we have no packets in the buffer; from mini-cycle $(a+1)$ to N for their duration we must also consider the waiting time in the buffer. Therefore, the duration Δ of the congestion avoidance phase is:

$$\Delta = aT + \frac{1}{\mu}\left[\left(\frac{N-1}{1-\lambda} + N - a + 1\right)(N-a)\right]$$

where a is the number of mini-cycles taken to saturate the bottleneck link :

$$a = \frac{1}{2}\left(\mu T - \frac{N-1}{1-\lambda}\right) + 1$$

References

1. S. Floyd, *A Report on Recent Developments in TCP Congestion Control*, IEEE Communications Magazine, Vol. 9, No. 4, April 2001.
2. L. S. Brakmo, L. L. Peterson, *TPC Vegas: end-to-end congestion avoidance on a global Internet*, IEEE JSAC,Vol.13, No.8, October 1995.
3. J. Mo, R. L. V. Anantharam, J. Walrand, *Analysis and comparison of TCP Reno and Vegas*, Proc. of IEEE Globecom'99, Rio de Janeiro (Brazil), December 1999.
4. Yuan-Cheng Lai, Chang-Li Yao, *The performance comparison between TCP Reno and TCP Vegas*, Proc. of Seventh International Conference on Parallel and Distributed Systems, Iwate (JAPAN), July 2000.
5. U. Hengartner, J. Bolliger and Th. Gross. *TCP Vegas Revisited*, Proc. of IEEE INFOCOM 2000, Tel Aviv (Israel), March 2000.
6. C. Fu et al., *Performance Degradation of TCP Vegas in Asymmetric Networks and its Remedies*, Proc. of ICC2001, Helsinki (Finland), June 11-14, 2001.
7. G. Hasegawa et al., *Analysis and Improvement of Fairness between TCP Reno and Vegas for Deployment of TCP Vegas to the Internet*, Proc. of ICNP, 2000.
8. R. W. Stevens, *TCP/IP Illustrated,Vol I The protocols*, Addison-Wesley, U.S.A., 1994.
9. T.V. Lakshman, U. Madhow, *The Performance of TCP/IP for Networks with High Bandwidth-Delay Product and Random Loss*, IEEE/ACM Transactions on Networking, Vol. 5, No. 3, June 1997.
10. Ait-Hellal, O.; Altman, E., *Analysis of TCP Vegas and TCP Reno*, Proc. of IEEE ICC '97, Vol. 1, Montreal (Canada), June 1997.
11. ns-LBL v.2.1b5, available via http://mash.cs.berkeley.edu/ns/ns.html.