Journal of Computer Science & Technology, Volume 19, Number 1, April 2019

- ORIGINAL ARTICLE -

Burst Error Analysis Introduced in Multiple Traffic of Protocols TCP Reno, Cubic, Westwood and Vegas on a Model of Hybrid Topology

Análisis de Error de Ráfaga Introducidos en un Tráfico Múltiple de los Protocolos TCP Reno, Cubic, Westwood y Vegas en un Modelo de Topología Híbrida

Diego R. Rodríguez Herlein[1], Carlos A. Talay[1], Claudia N. González[1], Franco A. Trinidad[1], Luz Almada[1] and Luis A. Marrone[2]

[1]*UARG, Universidad Nacional de la Patagonia Austral,Rio Gallegos, Santa Cruz, 9400, Argentina*
{dherlein, ctalay, cgonzalez}@uarg.unpa.edu.ar, talejandro.franco@gmail.com, mluzalmada@gmail.com
[2]*L.I.N.T.I., Universidad Nacional de La Plata,La Plata, Buenos Aires1900, Argentina*
lmarrone@linti.unlp.edu.ar

Abstract

This paper explores the behavior shown by the protocols TCP Reno, Cubic, Vegas and Westwood in presence of errors in bursts occurred in multiple traffic over a hybrid topology. The development is based on the analysis of case studies, where it starts with a mixed topology of two wired and two wireless nodes, with two flows of the same variant of the TCP protocol, subsequently increasing the number of nodes until reaching 8 wired nodes behaving as senders and 8 wireless nodes as receivers. In all cases only one of these flows suffers a burst error and through the tests we analyze how the flow recovers from the burst. For this study, behavioral tests were carried out using the NS-2 network simulator, on a hybrid topology (wired and wireless), also incorporating burst errors of different lengths, typical of wireless links.

Keywords: TCP, burst errors, NS-2

Resumen

Este artículo explora el comportamiento mostrado por los protocolos TCP Reno, Cubic, Vegas y Westwood en presencia de errores en ráfagas ocurridas en el tráfico múltiple sobre una topología híbrida. El desarrollo se basa en el análisis de estudios de caso, comenzando con una topología mixta de dos nodos cableados y dos inalámbricos, con dos flujos de una misma variante del protocolo TCP, aumentando el número de nodos hasta alcanzar 8 nodos cableados emisores y 8 nodos inalámbricos receptores. En todos los casos, solo uno de estos flujos sufre un error de ráfaga y, a través de las pruebas, analizamos cómo se recupera el flujo de la ráfaga. Para este estudio, las pruebas de comportamiento se llevaron a cabo utilizando el simulador de red NS-2, en una topología híbrida (cableada e inalámbrica), que también incorporó errores de ráfaga de diferentes longitudes, típicos de los enlaces inalámbricos.

Palabras claves: TCP, Errores en ráfaga, NS-2

1. Introduction

TCP (Transmission Control Protocol) [1] has been widely used in data networks since its development to the present. Although it was not part of the initial development, congestion control was one of the aggregates that significantly improved its performance. By means of modifications that try to adapt it to different working conditions, this protocol has accompanied the technological innovations that have been developed in the area of telecommunications and that use both wired and wireless media. TCP is characterized by being reliable, performing flow control and possessing a data congestion control mechanism. It also controls the sequence of segments delivery, by means of the verification of ordered reception of segmentssequence numbered in origin and verified in the receiver. This protocol offers a service oriented to connection, which bases its reliable delivery in a procedure known as ARQ (Automatic Repeat reQuest), in its different variants, which guarantees the integrity of the data. Through the ARQ procedure and the use of selective ACKs (acknowledgments), it is achieved that a whole set of segments can be confirmed with one ACK. This technique is known as delayed-ACK [2] and allows achieving a significant increase in efficiency in the operation of the network.

At the level of congestion control, TCP regulates traffic over the data flow. To achieve this, the protocol verifies if there is a loss of segments or if a reception of duplicate ACKs occurs. Analyzing the result of this verification, the protocol determines the occurrence of packet loss and therefore whether or not there is congestion in the network [3]. By enhancing this method, two variants have been developed to address congestion control problems. One of them is based on a reactive control of the problem, assuming that there is congestion in the links due to the loss of segments. On the other hand we have the other variant, which tries to perform a proactive congestion control, where it is developed a strategy to prevent traffic from reaching a situation of congestion; both are not mutually exclusive [4].

At present, transmission technologies point to the quality of the data flow, which make it possible to have a low error rate. In this scenario, the congestion control techniques of the network have been based mainly on the detection of lost segments. Therefore, under these conditions, the reactive protocols understand that there is congestion in the network and they activate their congestion control algorithms. However, there are situations in which that loss may have another origin than congestion and therefore should not trigger its mechanisms.

The growing expansion in the use of wireless networks created the need to modify the TCP protocol, originally designed for wired networks where congestion is the main cause of segment loss. TCP does not react adequately to segment losses unrelated to congestion: if there is a loss due to interference, then there are no overflows in the buffer and TCP decision to reduce the congestion window is incorrect, drastically reducing the performance. Instead, there must be a recovery from that loss and continue with the same rate as if nothing had happened.

Thinking about this situation, it is proposed by configuring a simple model and using the simulation tool NS-2 [5], to analyze the response of 4 TCP agents with the introduction of burst errors in a data transmission, without depending on any explicit notification of the network, preserving the host-to-host principle of TCP.

2. Theoretical framework

When the sending rate of all the TCPs sharing the same network exceeds its capacity, the effective load transported will tend to zero as the load increases. This effect was known as collapse of congestion [6].The original TCP standard lacks a means to adjust the transmission speed according to the state of the network. To solve this problem, several solutions have been proposed that share the same idea, that is, the introduction of a mechanism that limits the sending rate along with the flow control driven by the receiver. To this end, the concept of a congestion window was introduced, whose purpose is to estimate the amount of data that the network can accept for delivery without congestion.

One of the first host-to-host solutions [7] to solve the problem was TCP Tahoe [8]. The solution is based on the original TCP specification (RFC 793) and includes a series of algorithms to improve the detection of packet loss. The RTO (Retransmission Timeout) was originally defined as the only loss detection mechanism. Since the TCP receivers respond immediately to all the segment data out of order with a duplicate ACK, the loss can be detected by the Fast Retransmit algorithm [9], almost within the RTT (Round-trip time) interval, that is, that duplicate ACKs can be considered a reliable loss indicator. With this new indicator, the sender can retransmit lost data without waiting for the corresponding RTO event.

However, the most important incorporation was the mechanisms of Slow Start and Congestion Avoidance [10]. These provide two slightly distributed host-to-host mechanisms that allow the TCP sender to detect the available network resources and adjust the transmission speed.

In the Slow Start algorithm, the reception of an ACK segment increments the congestion window in one segment for each segment validated by the ACK(multiplicative increase policy). If a segment loss is detected, it is assumed that the network is under congestion, then the congestion window is reset to the initial value (eg. 1) to guarantee the release of network resources.

The Congestion Avoidance algorithm is aimed at improving the efficiency of TCP in networks with limited resources. This is a much more conservative algorithm, which increases the congestion window in only one segment if all the data segments have been delivered successfully during the last RTT. In contrast, the value of the congestion window is reduced by half (multiplicative reduction policy). TCP Tahoe includes both algorithms as distinct operational phases, which combines rapid discovery of network resources and long-term efficiency.

2.1 TCP Reno

Reducing the congestion window to a segment, as a reaction to a loss, can lead to significant performance degradation.

A completely different state of congestion can be inferred from a loss detected by RTO timer timeout, to that detected by the arrival of duplicate ACKs. The presence of each ACK, including duplicates, indicates the successful delivery of a data packet. The sender is observing the ability of the network to

deliver some data. Therefore, the state of the network can be considered slightly congested, and the reaction to the loss event may be more optimistic. TCP Reno achieves an optimistic reaction when using the Fast Recovery algorithm [11].

Fast Recovery halves the congestion window, and polls the network until the error is recovered and an unduplicated ACK is received.Compared to TCP Tahoe, the performance is substantially higher because the recovery period is reduced and data transfers are allowed during it.

2.2 TCP Cubic

This congestion control algorithm was designed to solve the problem in networks with High-BDP. TCP CUBIC [12] uses the approach to define the size of the congestion window as a cubic function of the time elapsed (Δ) since the last congestion event, also the size of the congestion window (wmax) just before the last detection of recorded loss and of a Beta coefficient that is a multiplicative decrease coefficient in FastRecovery.

$$w = C\,(\Delta - \sqrt[3]{beta2 - \frac{wmax}{C}})^3 + wmax \quad \text{(Eq. 1)}$$

Where, C is a predefined constant.The function has a very fast growth when the current window is far from the estimated target (size of the window before the previous loss), and it is very conservative when it is close.

2.3 TCP Vegas

Reno and Cubic share the same reactive method: it detects that the network is congested only if segments are lost. Reactive algorithms increase transmission rates to the extent that segment losses occur due to congestion, to find the capacity of the network.

Another approach, proactive method, is to quantify the level of congestion before a loss event occurs using a delay estimate of the segments.

Brakmo and Peterson proposed the Vegas algorithm as a proactive method to replace the reactive algorithm of Congestion Avoidance [13]

The key component is to estimate the use of the buffers by analyzing the RTT values. The minimum RTT value observed during the lifetime of the connection is considered a reference measurement that indicates a network status without congestion. In this way, a higher value of RTT is due to a greater length of the queue in the transmission path. The objective is to detect congestion at its early stage and avoid it by reducing its transmission rate, preventing the segment loss occurrence.

The technique consists of calculating the difference between the expected flow and the current data flow, in order to determine the remaining bandwidth in the network. The expected rate is a theoretical rate of a TCP flow in a network state without congestion. This speed can occur if all transmitted data segments are successfully recognized within the minimum RTT. The expected rate is directly proportional to the size of the congestion window with a proportionality coefficient of 1/RTTmin. The current rate can be expressed as the ratio between the current congestion window and the current RTT value.

TCP Vegas incorporates the measure of the difference between the current rate and the expected rate in the Congestion Avoidance phase to control the size of the congestion window. There is always a point where the current rate is equal to the expected rate, and all attempts to send at a faster rate will fail.

The concept of VegasTCP congestion control is that if the connection is congested, then the current flow rate is lower than the expected rate and the difference between them will indicate the degree of congestion, allowing adjusting the size of the congestion window. If this difference is greater than the predefined threshold Beta, the congestion window is reduced by one, if the difference is strictly less than a second Alpha threshold the size of the congestion window is increased by 1. If the difference is between Alpha and Beta, the system is considered to be in a stationary state and no modifications are applied to the congestion window, to avoid oscillations. In this way, Vegas uses these two thresholds, Alpha and Beta, to control the adjustment size of the congestion window [14].

In Slow Start mode, the size of the congestion window is doubled each time an RTT is completed. TCP Vegas modifies this algorithm so that it is able to detect and avoid congestion. While this allows window growth during this phase, it maintains its size while calculating the current and expected throughput. When the current throughput falls below the expected throughput by a certain amount (defined by a Gamma threshold), TCP Vegas changes to the Congestion Avoidance algorithm, where the window size is now adjusted in the following way [15]:

Calculation of the Expected data flow (*F. Expected*): size of the congestion window (cwnd) and the minimum measured value of RTT (*Base RTT*).

$$F.Expected = \frac{cwnd}{BaseRTT} \quad \text{(Eq. 2)}$$

Calculation of the current Data Flow (F. Current), size of the congestion window (cwnd) and RTT last measured value of RTT.

$$F.Current = \frac{cwnd}{RTT} \qquad \text{(Eq. 3)}$$

Then TCP Vegas compares the current performance with the expected performance and calculates the difference as diff (difference):

$$diff = \frac{F.\ Expected - F.\ Current}{Base\ RTT} \qquad \text{(Eq. 4)}$$

Based on these calculations, TCP Vegas adjusts the size of the congestion window in the following way:

$$\text{cwnd} = \begin{cases} \text{cwnd+1} ; & \text{if diff} < \alpha \\ \text{cwnd} ; & \text{if } \alpha \leq \text{diff} \leq \beta \\ \text{cwnd-1} ; & \text{if diff} > \beta \end{cases} \qquad \text{(Eq. 5)}$$

If segments are not lost in the network, Vegas controls the congestion window through an additive increase policy and additive reduction (AIAD) [16]. Reactions to segment losses are defined by any of the standard congestion control algorithms (for example those of Reno).

2.4 TCP Westwood

TCP Westwood replaces the congestion control actions used by TCP Reno, with a heuristic procedure to set the size of the congestion window to an optimal value. As an optimum, the heuristic considers a value that corresponds to a data transfer rate observed in the recent past. If a random error occurs, the optimal value will be that in which the sending TCP continues with the same sending rate. In the event that the loss of the segment is due to congestion, the rate at which data is received at the receiver is the rate at which the network can transport data. If the sender continues the transmission at a rate equal to that observed by the receiver, the number of newly transmitted segments will be equal to the number of segments delivered and, thus, the queues will not grow and additional congestion will be avoided.

The proposed solution is for the issuer to estimate the current delivery rate based on the existing notification mechanism (ACK).

If it is assumed that an ACK segment is generated immediately after receiving a data segment and that the ACKs are uniformly delayed in the return route, the ACK rate observed by the issuer will be equal to the data delivery rate observed by the receiver. The calculation of the bandwidth is maintained in the long term even if the receiver loses or delays some ACK; that is, a decrease in the ACK rate will be compensated for by an increase in the amount of recognized data.

To reduce the effects of fluctuation, Westwood calculates the bandwidth in two levels. At the first level, the estimate is calculated immediately after receiving an ACK segment by the amount of data recognized by the ACK and the time elapsed since the reception of the last ACK Δ.

$$b = \frac{d}{\Delta} \qquad \text{(Eq. 5)}$$

In the second level, the calculated instantaneous values are averaged with a special discrete-time filter:

$$B = \alpha(\Delta).B^{-1} + \left(1 - \alpha(\Delta)\right).\left(\frac{b+b^{-1}}{2}\right) \qquad \text{(Eq. 6)}$$

Where, α (Δ) is the average coefficient, as a function of Δ, b and b^{-1} are current and previous samples of the bandwidth estimate and B^{-1} is the previously calculated average value of the estimate.

3. The model used for the study and parameterization

To model and generate the data of the present work, the NS-2 (Network Simulator 2), simulator of networks of discrete events in its version ns-2.35 (released Nov. 4 2011) was used and the following topology was implemented:

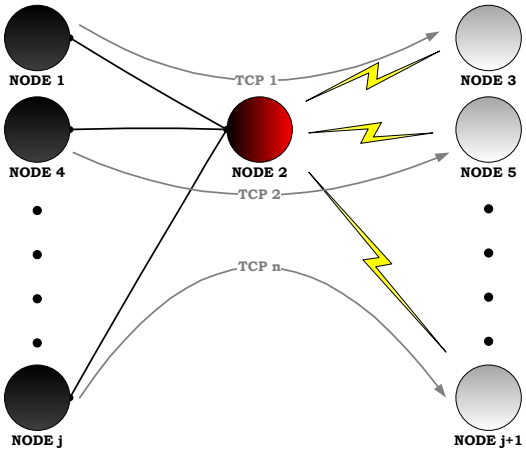

Fig.1 Schematic of the topology for the tests

As shown in Figure 1, nodes 1 and 2 are linked by a wired link that was configured as duplex, with a bandwidth of 2 Mb / s, propagation delay 2 ms. and DropTail queue service policy. The link between nodes 2 and 3 is wireless and was configured as TwoRayGround propagation mode, the WirelessPhy physical layer, MAC 802.11, the OmniAntenna antenna and the wireless node without mobility.

The selection of this model is an approximation to a Wireless scenario with a fixed node (node 1), a base station (node 2) and a mobile node (node 3), with practical simplification, that the wireless link does not present disconnections and it only has errors in the form of bursts. In principle the model consisted of more wireless nodes, but because the present work was based on the study of errors in the

form of bursts in hybrid media, it was not considered for the moment to increase the complexity considering a larger number of mobile nodes.

Node 1 was configured as a sender and in it a TCP agent, on the other hand node 3 was configured as a receiver. This link was associated with an FTP traffic (file transfer protocol) as the only traffic.

Independent simulations were carried out on the implementations of the different variants of TCP. For each of them, simulations were generated for the different lengths of errors in bursts, with lengths ranging from a test without errors (0), to tests with bursts of error of 5, 10, 15 and 20 segments of 1K length. The TCP Agents that were used were Reno, Cubic, Vegas, and Westwood, as they are designated and implemented in this version of NS-2 (see 2.35), without any modification. In the case of TCP Vegas, for Alpha and Beta, values that have been implemented in NS-2, Alpha= 1 and Beta= 3 are used by default.

The data transmission begins 5 seconds after the simulation starts and is conditioned to the transmission of 3,000 segments of 1,000 bytes each, regardless of the length of the error burst. These bursts always started after the first 999 segments were transmitted and the test concluded upon completion of transmitting the 3,000 segments of FTP traffic.

Based on what was analyzed in the paper presented at the CACIC 2017 [17], a new trial was proposed modifying the topology of the network and making the model more complex by adding a wired node and a wireless node in order to add a new TCP flow of the same variant, between the new nodes. The new nodes and links have the same characteristics as the original nodes and links and the new TCP flow is an FTP that has no errors and transmits the number of segments necessary to be transmitting until the end of the simulation.

This procedure was repeated successively in different simulations for each of the variants of TCP, for each of the lengths of errors in stipulated bursts adding two nodes and a TCP flow until arriving at the 8 simultaneous FTP transferences, where only the first presents errors in burst.

An AWK script was used on the trace file to obtain instantaneous and average Throughput, in all cases of the first TCP flow. These data were processed, turned over to a spreadsheet and generated the graphs presented here.

4. Results obtained

With this test, based on a bunch of simulations, it is intended to determine the throughput behavior of the four proposed protocols in the presence of burst errors of different segment lengths.

These simulations and results are based on the work "Considerations on the behavior TCP protocol in its variants Vegas, Reno, Cubic and Westwood before errors in burst generated in a hybrid topology" published in the XXIII Argentine Congress of Computer Science (CACIC 2017, La Plata, Argentina) and presented at the XII Workshop on Architectures, Networks and Operative Systems (WARSO) in October2017.

Below are the graphs of the simulations obtained throughput vs. time and sequence number vs. time, superimposing the results for the case of having 1, 2, 4, and 8 nodes, with their corresponding established individual traffics as explained above, for the same variant of the TCP protocol. The graphs have been grouped taking into account the base case of a simulation without errors and a posteriori tests for a progression of burst errors that vary according to 5, 10, 15 and 20 lost segments.

4.1 Tests with TCP Reno

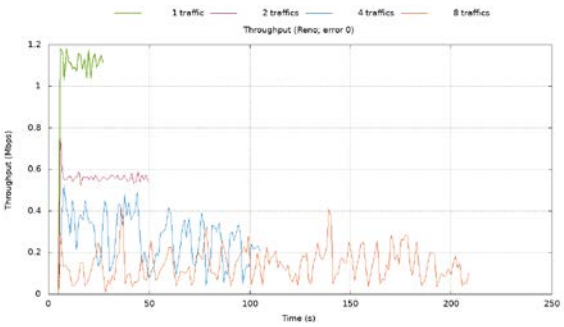

Fig. 2 Throughput vs. Time – without errors

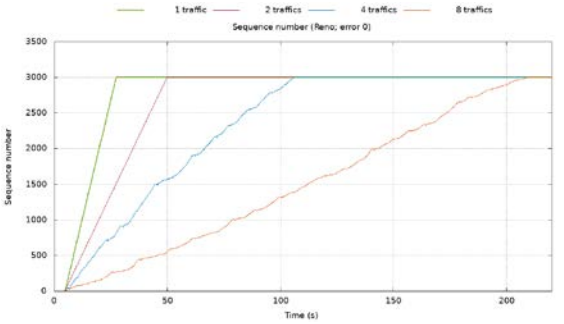

Fig. 3 Sequence Nº vs. Time – without errors

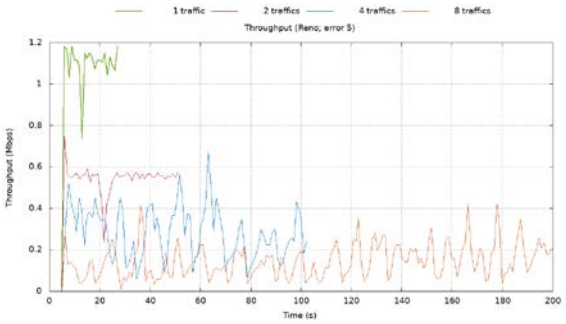

Fig. 4 Throughput vs. Time – 5 errors

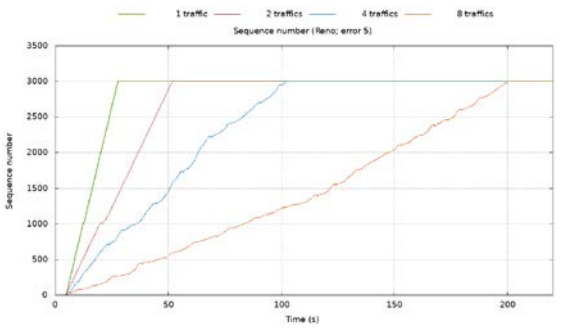

Fig. 5 Sequence Nº vs. Time – 5 errors

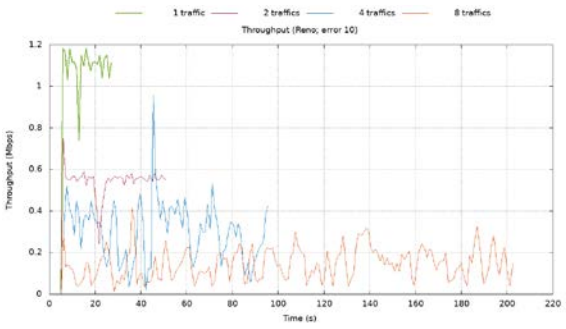

Fig. 6 Throughput vs. Time – 10 errors

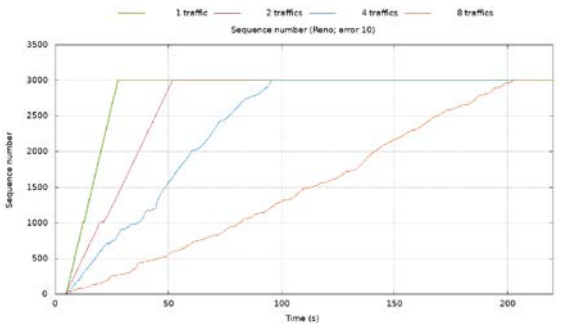

Fig. 7 Sequence Nº vs. Time – 10 errors

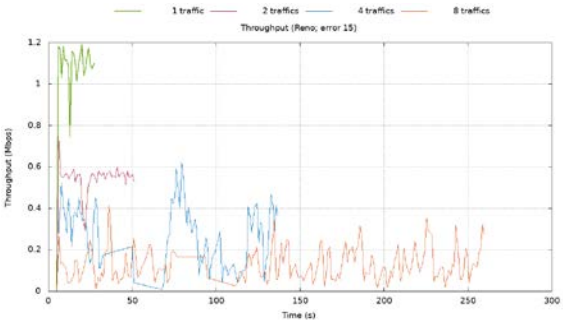

Fig. 8 Throughput vs. Time – 15errors

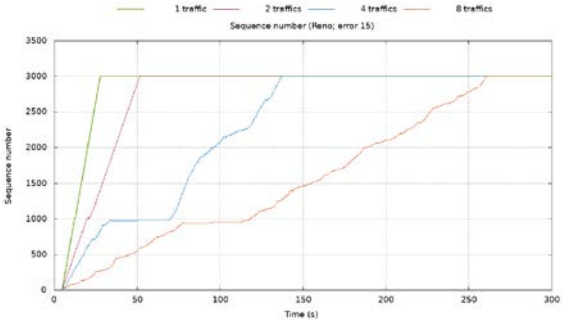

Fig. 9 Sequence Nº vs. Time – 15 errors

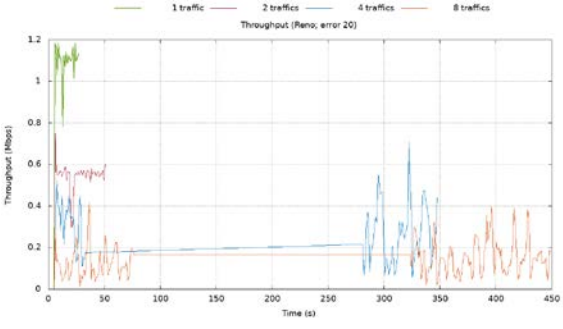

Fig. 10 Throughput vs. Time – 20 errors

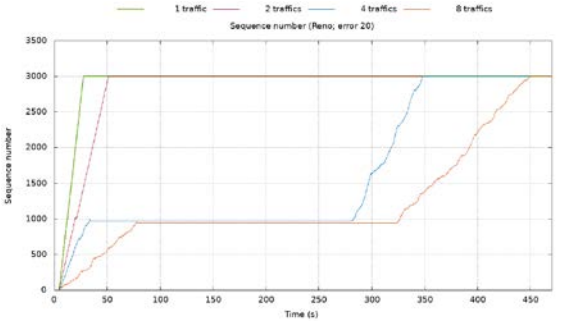

Fig. 11 Sequence Nº vs. Time – 20 errors

In the tests carried out for TCP Reno, the traffic of 4 and 8 flows are affected from the 15 burst errors. The increase of time in the transmission sequence responds to a typical sequence analysis with a "stretch" of time. In particular, there is little

difference between the delay in the transmission suffered when we use 4 and 8 simultaneous traffics.

4.2 Tests with TCP Cubic

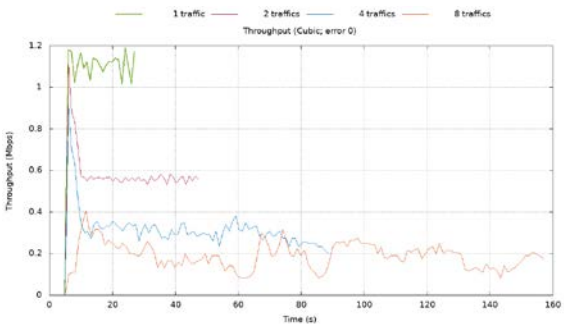

Fig. 12 Throughput vs. Time – without errors

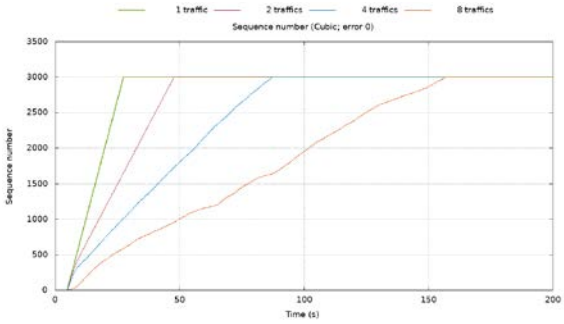

Fig. 13 Sequence Nº vs. Time – without errors

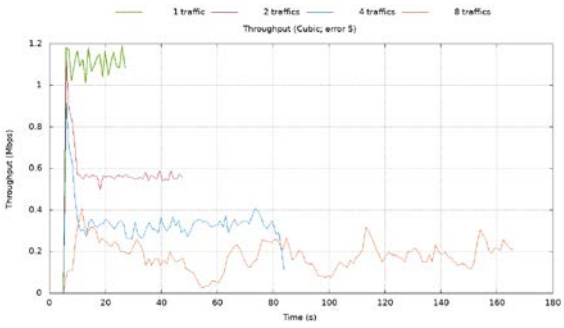

Fig. 14 Throughput vs. Time – 5 errors

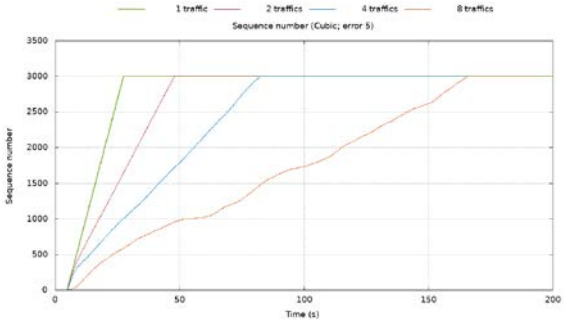

Fig. 15 Sequence Nº vs. Time – 5 errors

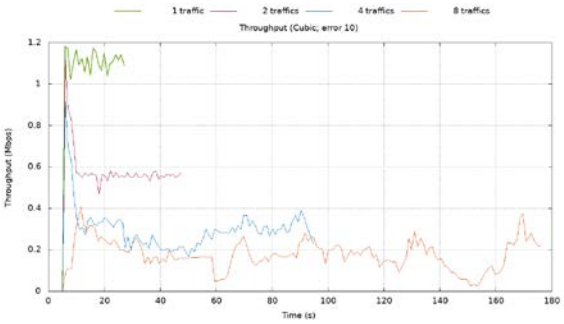

Fig. 16 Throughput vs. Time – 10errors

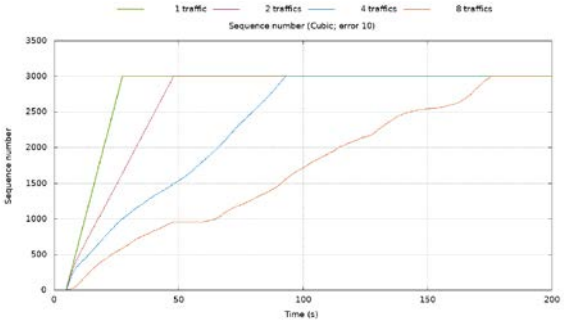

Fig. 17 Sequence Nº vs. Time – 10 errors

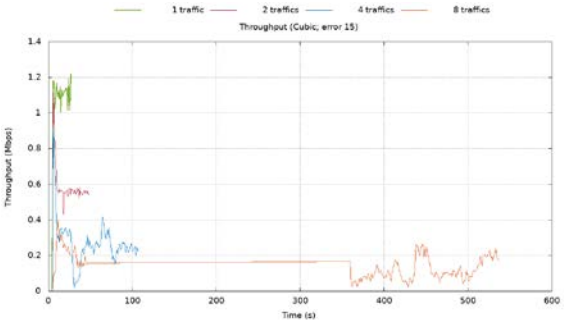

Fig. 18 Throughput vs. Time – 15errors

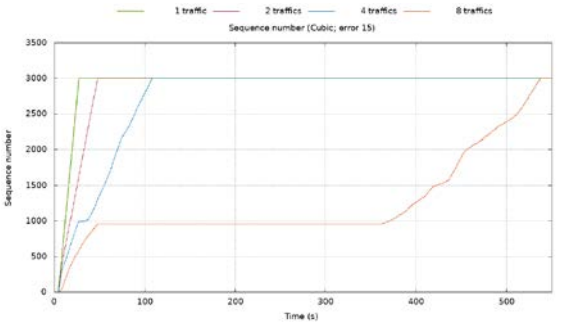

Fig. 19 Sequence Nº vs. Time – 15 errors

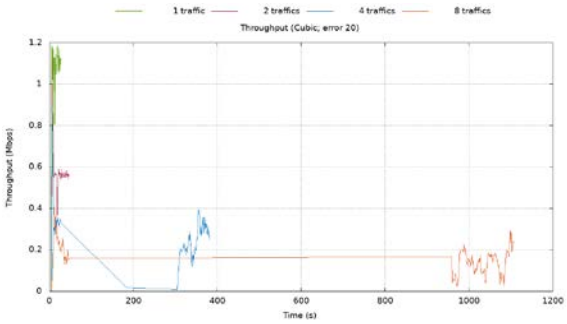

Fig. 20 Throughput vs. Time – 20 errors

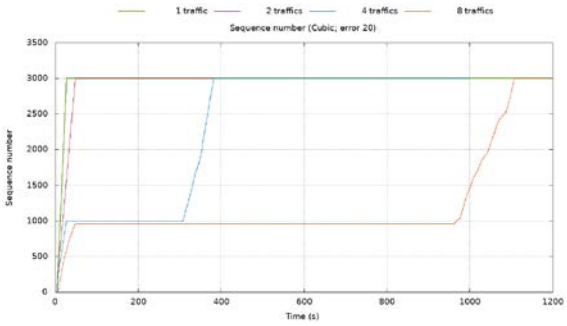

Fig. 21 Sequence Nº vs. Time – 20 errors

TCP CUBIC presents an early sensitivity to burst errors. For the trials of 10 burst errors, CUBIC already shows symptoms of being affected; this can be seen in the graph of packet sequence and throughput vs. time. In both cases, for T= 45 the transmission is delayed for approximately 15 seconds, recovering after that time the data transmission. For larger burst error values, the delay in recovering the transmission is noticeably higher.

4.3 Tests with TCP Westwood

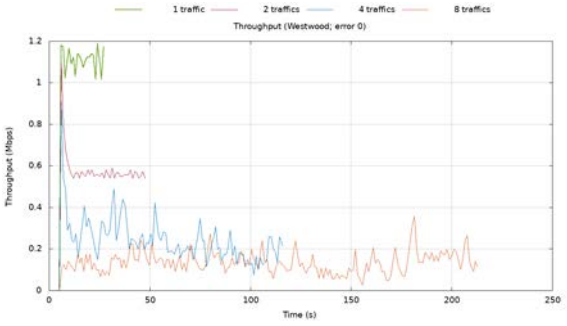

Fig. 22 Throughput vs. Time – no errors

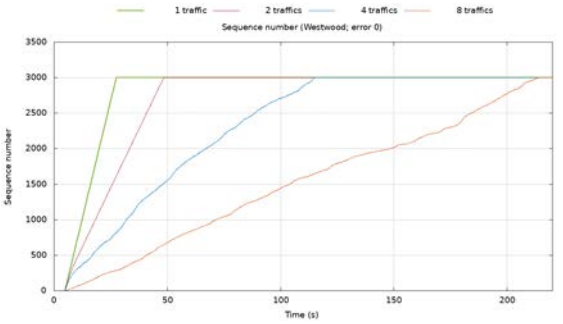

Fig. 23 Sequence Nº vs. Time – no errors

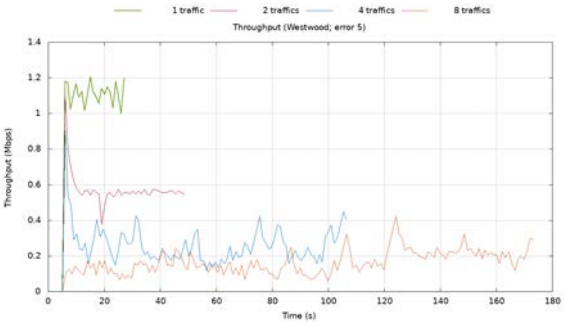

Fig. 24 Throughput vs. Time – 5 errors

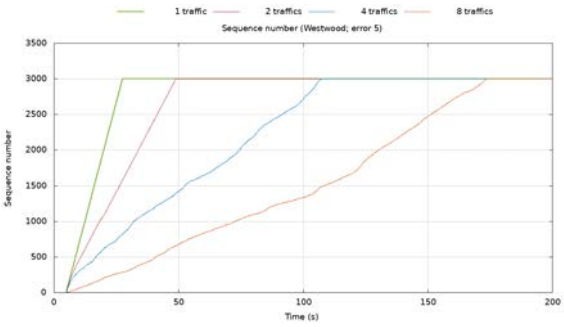

Fig. 25 Sequence Nº vs. Time – 5 errors

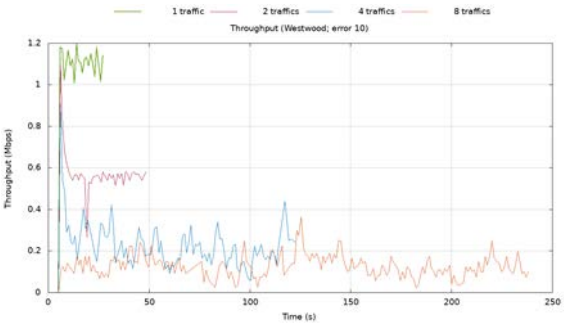

Fig. 26 Throughput vs. Time – 10 errors

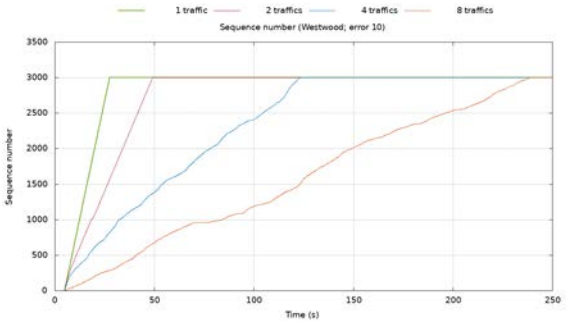

Fig. 27 Sequence Nº vs. Time – 10 errors

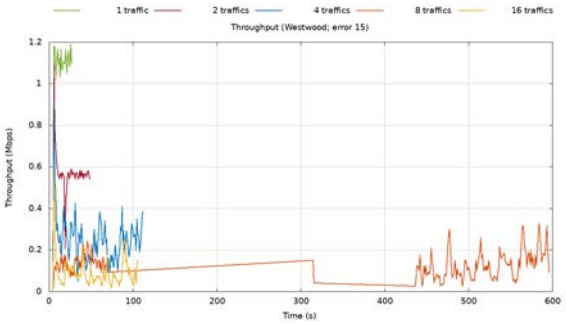

Fig. 28 Throughput vs. Time – 15 errors

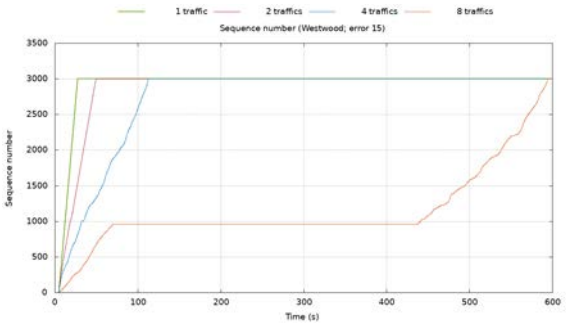

Fig. 29 Sequence Nº vs. Time – 15 errors

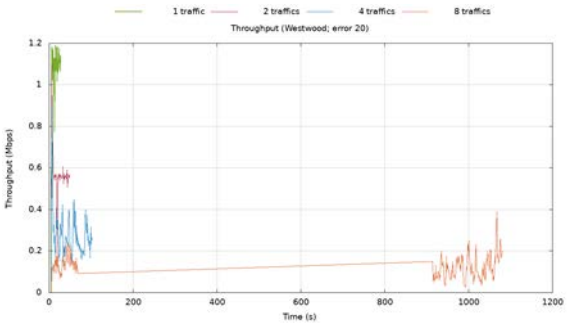

Fig. 30 Throughput vs. Time – 20 errors

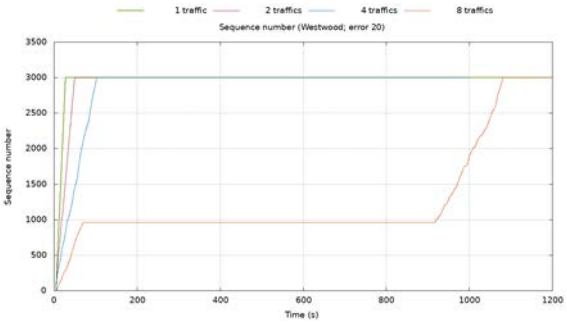

Fig. 31 Sequence Nº vs. Time – 20 errors

As we can see, Westwood has a great sensitivity to burst errors from the 15 packets and 8 simultaneous traffics. In that test and posteriors of 20 packets of error in burst, particularly the sequence of packages presents a remarkable "stretching" which indicates that the transfer of data is suspended for a while until resuming the sending of data.

4.4 Tests with TCP Vegas(Alpha=1;Beta=3)

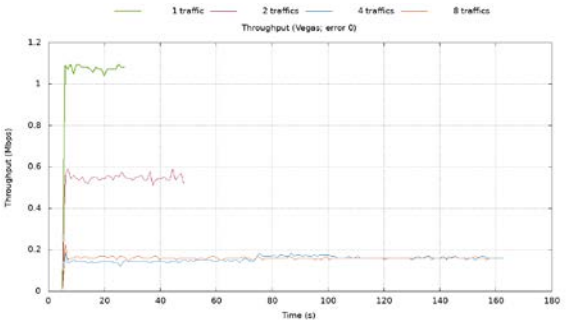

Fig. 32 Throughput vs. Time – without errors

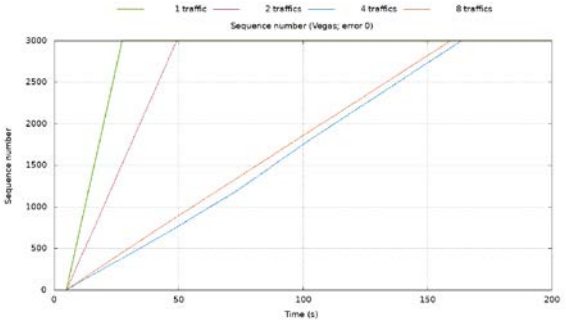

Fig. 33 Sequence Nº vs. Time – without errors

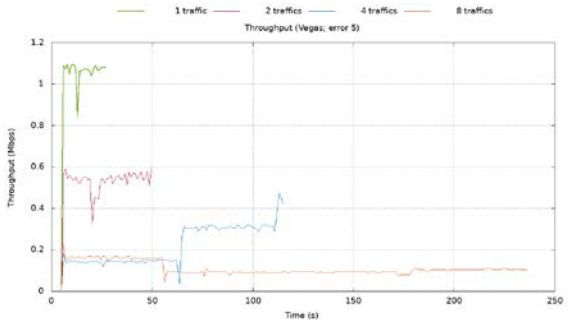

Fig. 34 Throughput vs. Time – 5 errors

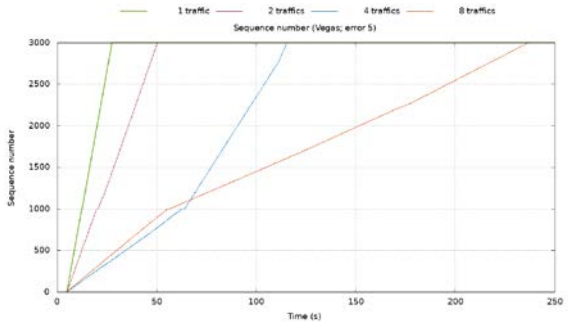

Fig. 35 Sequence Nº vs. Time – 5 errors

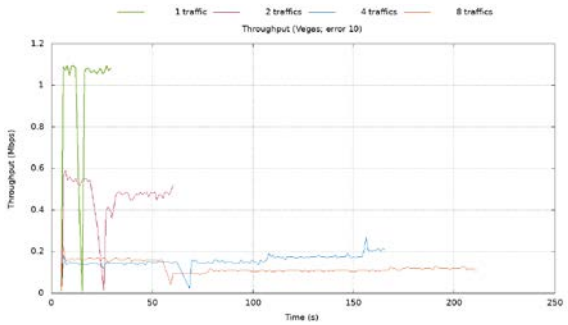

Fig. 36 Throughput vs. Time – 10 errors

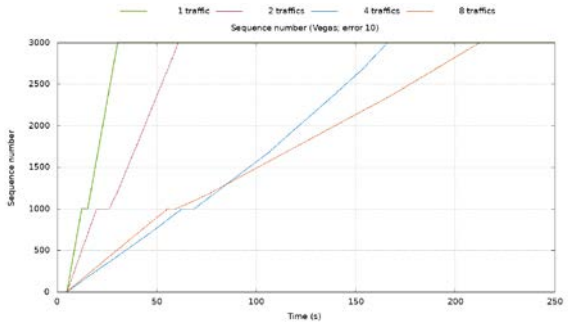

Fig. 37 Sequence Nº vs. Time – 10 errors

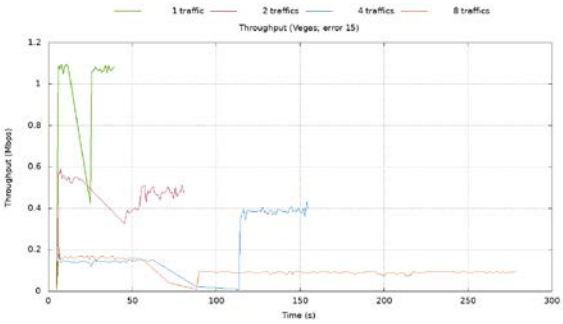

Fig. 38 Throughput vs. Time – 15 errors

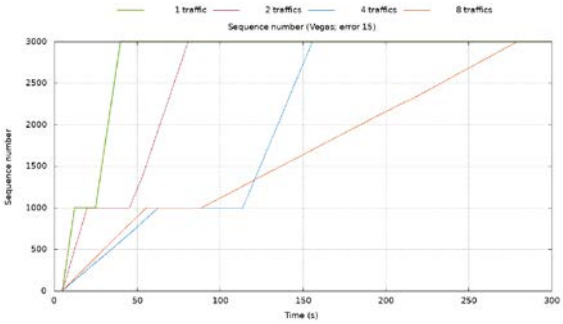

Fig. 39 Sequence Nº vs. Time – 15 errors

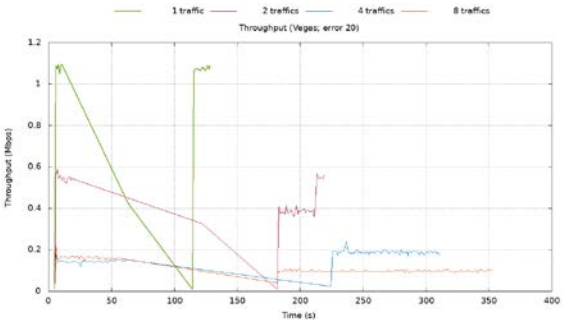

Fig. 40 Throughput vs. Time – 20 errors

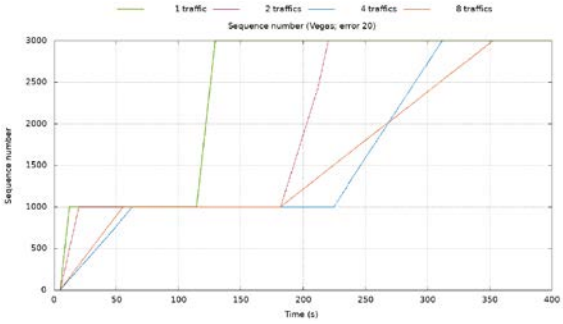

Fig. 41 Sequence Nº vs. Time – 20 errors

4.5 Tests with TCP Vegas (Alpha=4;Beta=8)

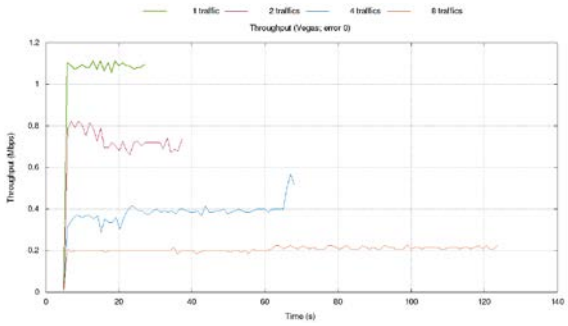

Fig. 42 Throughput vs. Time – without errors

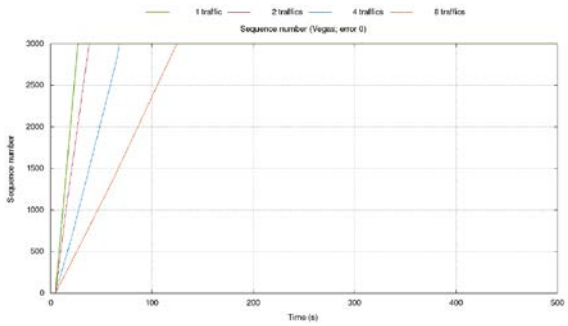

Fig. 43 Sequence Nº vs. Time – without errors

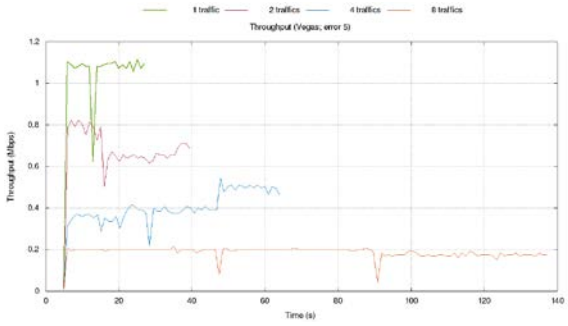

Fig. 44 Throughput vs. Time – 5 errors

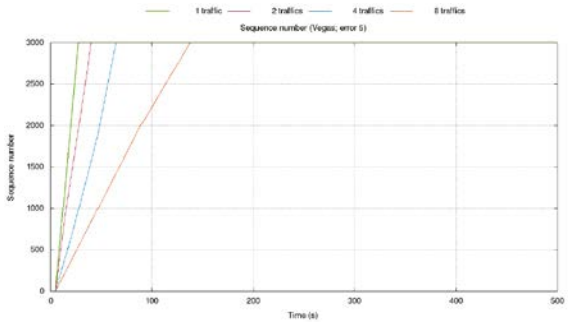

Fig. 45 Sequence Nº vs. Time – 5 errors

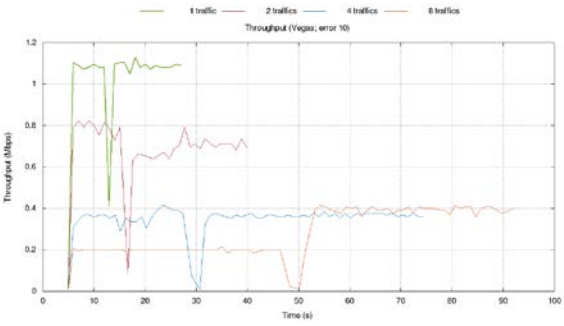

Fig. 46 Throughput vs. Time – 10 errors

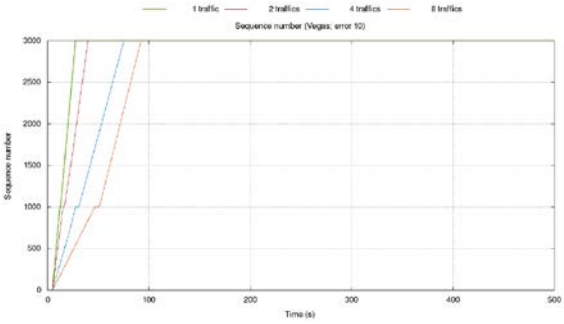

Fig. 47 Sequence Nº vs. Time – 15 errors

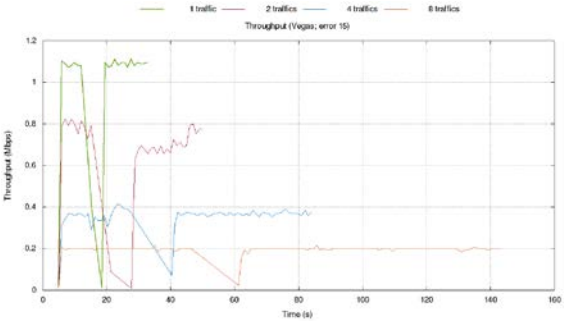

Fig. 48 Throughput vs. Time – 15 errors

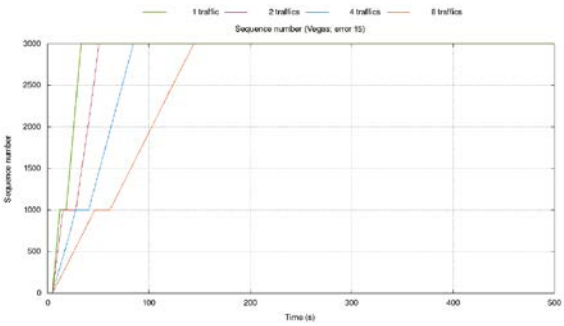

Fig. 49 Sequence Nº vs. Time – 15 errors

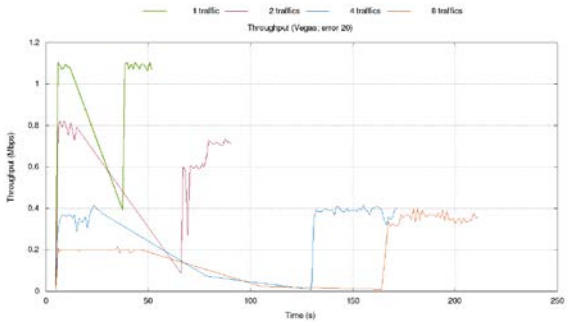

Fig. 50 Throughput vs. Time – 20 errors

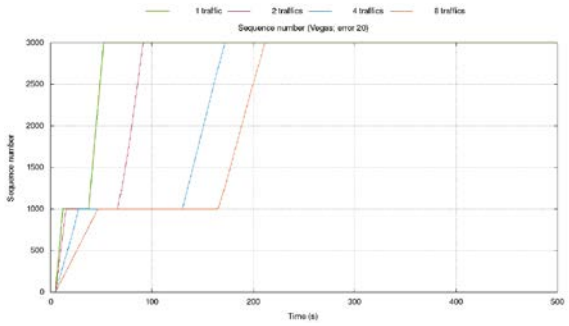

Fig. 51 Sequence Nº vs. Time – 20 errors

As we can see, although TCP Vegas is affected in a similar way for all tests of different flows (1, 2, 4 and 8) equally for alpha= 1, beta= 3 and alpha= 4 beta= 8, in this last case the transmission sequence offers correspondence between different traffic values. Particularly the case of Vegas (alpha= 4, beta= 8) is the protocol least sensitive to burst errors with a shorter recovery time than all previous tests.

5. Conclusions

As observed in the results, as the size of the burst and the traffic increases, the recovery time in the transmission of data for the flow where the burst error is generated, increases. This effect is verified in both the throughput vs. time as well as the sequence number vs. time graphics. In particular, we can note that on the tests performed, this effect becomes more noticeable from a burst size of 15 segments, accentuated when we perform tests by increasing the amount of flows in a common path, which implies greater competition for the bandwidth.

Of the protocols analyzed, TCP Vegas is the one that shows particular characteristics. In the first test sequence, the default parameters were maintained using NS-2 with alpha= 1 and beta= 2, figures 32 to 41. On this set, the graphs representing the sequence number vs. time of the flow where the burst error is found (figures 35, 37, 39 and 41), we see that at times when the traffic is in a set of 4 simultaneous data streams, the transmission times are greater than when it is in a set of 8 simultaneous data streams. In all cases, the trend is reversed and at the end of the test, the traffic that is in a set of 4 simultaneous flows ends before the transmission of the 3,000 segments that the traffic that is in a set of 8 flows. This generates that at any given moment the celestial and red curves cross.

This effect does not occur when the tests are carried out with parameters of alpha= 4 and beta= 8, in all cases, the greater the number of data flows considered, the greater the transmission times of the data traffic with burst errors.

6. Future research lines

For future work, it is proposed to study other variants of the TCP protocol in similar conditions to try to determine a behavior in the response to these conditions.

Competing interests

The authors have declared that no competing interests exist.

References

[1] Postel J., “RFC 793: Transmission Control Protocol”.September 1981.
[2] David Clark, “RFC 813: Window and Acknowledgment Strategy in TCP”. July 1982
[3] M. Handley, J. Padhye and S. Floyd, “TCP Congestion Window Validation”. RFC 2861, June 2000.
[4] A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock, “Host-to-Host Congestion Control for TCP”. IEEE Communications Surveys Tutorials, vol. 12, no. 3, 3rd quarter 2010, pp. 304- 340.
[5] Teerawat, Issariyakul & Ekram Hossain, “Introduction to Network Simulator NS2”. Springer, 2009
[6] J. Nagle, “RFC896—Congestion control in IP/TCP internetworks”. 1984
[7] A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock,“Host-to-Host Congestion Control for TCP”. IEEE Communications Surveys & Tutorials, Volume 12 Issue, page 304-342.3, July 2010
[8] V. Jacobson, “Congestion avoidance and control”. ACM SIGCOMM, pp. 314–329, 1988.
[9] R. Braden, “RFC1122—Requirements for Internet Hosts – Communication Layers”. 1989.
[10] W. Stevens, “RFC2001—TCP Slow Start, Congestion Avoidance, Fast Retransmit”. 1997.
[11] V. Jacobson, “Modified TCP congestion avoidance algorithm”. Technical report, April 1990.

[12] I. Rhee and L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant". SIGOPS Operating Systems Review, vol. 42, no. 5, pp. 64–74, July 2008.
[13] L. Brakmo and L. Peterson, "TCP Vegas: end to end congestion avoidance on a global Internet". IEEE Journal Selection Areas Communications, vol. 13, no. 8, pp. 1465–1480, October 1995
[14] Tsang, E. C. M., Chang, R .K. C., "A Simulation Study on the Throughput Fairness of TCP Vegas", in Proceeding of 9th IEEE International Conference on Networks, pp. 469-474, Bangkok, Thailand, 2001
[15] Ghassan A. A., Mahamod I. &Kasmiran J., "Influence of Parameters Variation of TCP-Vegas in Performance of Congestion Window over Large Bandwidth-Delay Networks", in 17th Asia-Pacific Conference on Communications (APCC), pp. 434-438, Sabah, Malaysia, 2011.
[16] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang, "TCP Westwood: Bandwidth estimation for enhanced transport over wireless links". In Proc. ACM MOBICOM, pp. 287–297, 2001.
[17] D. R. Rodríguez Herlein, C. A. Talay, C. N. González, L. A. Marrone, F. A. Trinidad, "Consideraciones sobre el comportamiento del protocolo TCP en sus variantes Vegas, Reno, Cubic y Westwood ante errores en ráfaga en una topología híbrida", en XXIII Congreso Argentino de Ciencias de la Computación, pp. 874-883, La Plata, Oct. 2017

Citation: D.R. Rodríguez Herlein, C.A. Talay, C.N. González, F.A. Trinidad, L. Almada and L.A. Marrone. "*Burst Error Analysis Introduced in Multiple Traffic of Protocols TCP Reno, Cubic, Westwood and Vegas on a Model of Hybrid Topology"*, Journal of Computer Science & Technology, vol. 19, no. 1, pp. 32–44, 2019.

DOI: 10.24215/16666038.19.e04

Received: December 19, 2017 **Accepted:** October 16, 2018.