The Fluid Flow Approximation of the TCP Vegas and Reno Congestion Control Mechanism

Adam Domański[2], Joanna Domańska[1], Michele Pagano[3],
and Tadeusz Czachórski[1(✉)]

[1] Institute of Theoretical and Applied Informatics, Polish Academy of Sciences,
Baltycka 5, 44–100 Gliwice, Poland
tadek@iitis.pl
[2] Institute of Informatics, Silesian Technical University,
Akademicka 16, 44–100 Gliwice, Poland
adamd@polsl.pl
[3] Department of Information Engineering, University of Pisa,
Via Caruso 16, 56122 Pisa, Italy
m.pagano@iet.unipi.it

Abstract. TCP congestion control algorithms have been design to improve Internet transmission performance and stability. In recent years the classic Tahoe/Reno/NewReno TCP congestion control, based on losses as congestion indicators, has been improved and many congestion control algorithms have been proposed. In this paper the performance of standard TCP NewReno algorithm is compared to the performance of TCP Vegas, which tries to *avoid* congestion by reducing the congestion window (CWND) size before packets are lost. The article uses fluid flow approximation to investigate the influence of the two above-mentioned TCP congestion control mechanisms on CWND evolution, packet loss probability, queue length and its variability. Obtained results show that TCP Vegas is a fair algorithm, however it has problems with the use of available bandwidth.

1 Introduction

In spite of the rise of new streaming applications and P2P protocols that try to avoid traffic shaping techniques and the definition of new transport protocols such as DCCP, TCP still carries the vast majority of traffic [10] and so its performance highly influences the general behavior of the Internet. Hence, a lot of research work has been done to improve TCP and, in particular, its congestion control features.

The first congestion control rules were proposed by Van Jacobson in the late 1980s [8] after that the Internet had the first of what became a series of congestion collapses (sudden factor-of-thousand drop in bandwidth). The first practical implementation of TCP congestion control is known as TCP Tahoe, while further evolutions are TCP Reno and TCP NewReno that better handles multiple losses in the same congestion window (CWND). The Reno/NewReno algorithm

consists of the following mechanisms: Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery. The first two, determining an exponential and linear grow respectively, are responsible for increasing CWND in absence of losses in order to make use of all the available bandwidth. Congestion is detected by packet losses, which can be identified through timeouts or duplicate acknowledgements (Fast Retransmit). Since the latter are associated to mild congestion, CWND is just halved (Fast Recovery) and not reduced to 1 packet as after a timeout. Hence, the core of classical TCP congestion control is the AIMD (Additive-Increase/Multiplicative-Decrease) paradigm. Note that this approach provides congestion control, but does not guarantee fairness [6].

The TCP Vegas was the first attempt of a completely different approach to bandwidth management and is based on congestion detection *before* packet losses [3]. In a nutshell (see Sect. 2 for more details), TCP Vegas compares the *expected rate* with the *actual rate* and uses the difference as an additional congestion indicator, updating CWND to keep the *actual rate* close to the *expected rate* and, at the same time, to be able of making use of newly available channel capacity. To this aim TCP Vegas introduces two thresholds (α and β), which trigger an Additive-Increase/Additive-Decrease paradigm in addition to standard AIMD TCP behavior. The article [12] shows TCP Vegas stability and congestion control ability, but, in competition with AIMD mechanism, it cannot fully use the available bandwidth.

The goal of our paper is to compare the performance of these two variants of TCP through fluid flow models. In more detail we investigated the influence of these two TCP variants on CWND changes and queue length evolution, hence also one-way delay and its variability (jitter). Moreover, we also evaluated the friendliness and fairness of the different TCP variants as well as their ability in using the available bandwidth in presence of both standard FIFO queues with tail drop and Active Queue Management (AQM) mechanisms in the routers.

Another important contribution of our work is that we considered also the presence of background traffic and asynchronous flows. In the literature, traffic composed of TCP and UDP streams has been already considered, but in most works (for instance, in [5,13]) all TCP sources had the same window dynamics and UDP streams were permanently associated with the TCP stream. Instead, in this paper, extending our previous work presented in [4], the TCP and UDP streams are treated as separate flows. Moreover, unlike [9] and [14], TCP connections start at different times with various values of initial CWND.

The rest of the paper is organized as follows. The fluid flow approximation models are presented in Sect. 2, while Sect. 3 discusses the comparison results. Finally, Sect. 4 ends the paper with some final remarks.

2   Fluid Flow Model of TCP NewReno and Vegas Algorithms

This section presents two fluid flow models of a TCP connection, based on [7,11] (TCP NewReno) and [2] (TCP Vegas). Both models use fluid flow approximation

and stochastic differential equation analysis. The models ignore the TCP timeout mechanisms and allow to obtain the average value of key network variables.

In [11] a differential equation-based fluid model was presented to enable transient analysis of TCP Reno/AQM networks (flows throughput and queues length in bottleneck router). The authors also showed how to obtain ordinary differential equations by taking expectations of the stochastic differential equations and how to solve the resultant coupled ordinary differential equations numerically to get the mean behavior of the system. In more detail, the dynamics of the TCP window for the i-th stream are approximated by the following equation [7]:

$$\frac{dW_i(t)}{dt} = \frac{1}{R_i(t)} - \frac{W_i(t)W_i(t - R_i(t))}{2R_i(t - R_i(t))}p(t - R_i(t)) \tag{1}$$

where:

– $W_i(t)$ – expected size of CWND (packets),
– $R_i(t) = \frac{q(t)}{C} + T_{p_i}$ – RTT (sec),
– $q(t)$ – queue length (packets),
– C – link capacity (packets/sec),
– T_{p_i} – propagation delay (sec),
– p – probability of packet drop.

The first term on the right hand side of the Eq. (1) represents the rate of increase of CWND due to incoming acknowledgments, while the second one models multiplicative decrease due to packet losses. Note that such model ignores the slow start phase as well as packet losses due to timeouts (a loss just halves the congestion window size) in accordance with a pure AIMD behavior, which is a good approximation of the real TCP behavior in case of low loss rates.

In solving Eq. (1) it is also necessary to take into account that the maximum values of q and W depend on the buffer capacity and the maximum window size (if the scale option is not used, 64 KB due to the limitation of the AdvertisedWindow field in TCP header). The dropping probability $p(t)$ depends on the discarding algorithm implemented in the routers (AQM vs. tail drop) and on the current queue size $q(t)$, which can be calculated through the following differential equation (valid for both models also in presence of background UDP traffic):

$$\frac{dq(t)}{dt} = \sum_{i=1}^{n_1} \frac{W_i(t)}{R_i(t)} + \sum_{i=1}^{n_2} U_i(t) - C\mathbf{1}_{q(t)>0} \tag{2}$$

where $U_i(t)$ is the rate of the i-th UDP flow (with $U_i(t) = 0$ before the source starts sending packets), while n_1 and n_2 denote the number of TCP (NewReno or Vegas) and UDP streams, respectively. Note that the indicator function $\mathbf{1}_{q(t)>0}$ takes into account that packets are drawn at rate C only when the queue is not empty.

As already mentioned, classical TCP variants base their action on the detection of losses. The TCP Vegas mechanism, instead, tries to estimate the available bandwidth on the basis of changes in RTT and, every RTT, increases or decreases

CWND by 1 packet. To this aim, TCP Vegas calculates the minimum value of the RTT, denoted as R_{Base} in the following, assuming that it is achieved when only one packet is enqueued:

$$R_{Base} = \frac{1}{c} + T_p \tag{3}$$

Hence, the *expected rate*, which denotes the target transmission speed, is the ratio between CWND and the minimum RTT, i.e.:

$$Expected = \frac{W_i(t)}{R_{Base}} \tag{4}$$

while the *actual rate* depends on the current value $R(t)$ of the RTT:

$$Actual = \frac{W_i(t)}{R(t)} = \frac{W_i(t)}{\frac{q(t)}{c} + T_p} \tag{5}$$

The Vegas mechanism is based on three thresholds: α, β and γ, where α and β refer to the Additive-Increase/Additive-Decrease paradigm, while γ is related to the modified slow-start phase [3].

In more detail, for $Expected - Actual \leq \frac{\gamma}{R_{Base}}$ TCP Vegas is in the slow start phase, while for higher values of the difference we have the pure additive behavior: for $Expected - Actual \leq \frac{\alpha}{R_{Base}}$ the window increases by one packet for each RTT and for $Expected - Actual \geq \frac{\beta}{R_{Base}}$ the window decreases by the same amount. Finally, if $Expected - Actual$ is between the two thresholds α and β, CWND is not changed. Taking into account the definition of *expected* and *actual* rates given by Eqs. (4) and (5) respectively, it is possible to express the previous inequalities in terms of W_i, R and R_{Base}. Then, changes in the window are given by the formula:

$$\frac{dW_i(t)}{dt} = \frac{W(t - R(t)) * W(t - R(t))}{R(t)} p_0(t - R(t)) + \frac{1}{R(t - R(t))} p_1(t - R(t))$$
$$- \frac{1}{W(t - R(t))R(t - R(t))} p_2(t - R(t))$$

where

$$p_0 = \begin{cases} 1 & \text{for } \frac{W_i(R - R_{Base})}{R} \leq \gamma \\ 0 & \text{otherwise} \end{cases}, \qquad p_1 = \begin{cases} 1 & \text{for } \gamma \leq \frac{W_i(R - R_{Base})}{R} \leq \alpha \\ 0 & \text{otherwise} \end{cases}$$

and

$$p_2 = \begin{cases} 1 & \text{for } \frac{W_i(R - R_{Base})}{R} \geq \beta \\ 0 & \text{otherwise} \end{cases}.$$

3   Experimental Results

Our main goal is to show the behavior of the two completely different TCP mechanisms, taking into account various network scenarios in terms of amount of TCP flows as well as queue management disciplines (namely, standard FIFO with tail drop and RED, the best-known example of AQM mechanism). For numerical fluid flow computations we used a software written in Python and previously presented in [4]. During the tests we assumed the following TCP connection parameters:

– transmission capacity of bottleneck router: $C = 0.075$,
– propagation delay for i-th flow: $T_{p_i} = 2$,
– initial congestion window size for i-th flow (measured in packets): $W_i = 1, 2, 3, 4....$,
– starting time for i-th flow
– threshold values in TCP Vegas sources: $\gamma = 1$, $\alpha = 2$ and $\beta = 4$ (see [1,3]),
– RED parameters: $Min_{th} = 10$, $Max_{th} = 15$, buffer size = 20 (all measured in packets), $P_{max} = 0.1$, weight parameter $w = 0.007$,
– the number of packets sent by i-th flow (finite size connections).

Figures 1 and 2 present the CWND evolution and the buffer occupancy for different numbers of TCP Vegas connections. In more detail, Fig. 1(a) refers to a single TCP stream: after the initial slow start, the congestion avoidance phase goes on until the optimal window size is reached and then CWND is maintained at such level until the end of transmission. In case of two TCP connections (Fig. 1(b)), the evolution of CWND is identical for both streams and similar to the single source case (apart from a slightly lower value of the maximum CWND). The comparison between the two figures highlights the main disadvantage of TCP Vegas: the link underutilization. Indeed, under the same network conditions, the optimal CWND for one flow is only slightly less than the optimal CWND for each of the two flows.
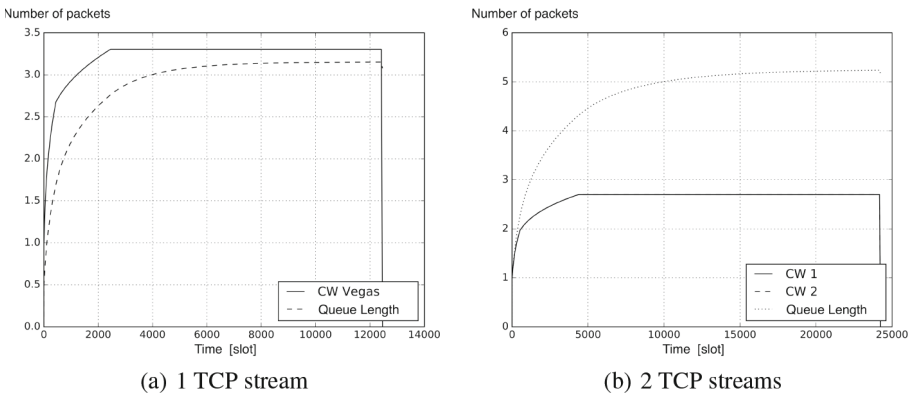

(a) 1 TCP stream   (b) 2 TCP streams

Fig. 1. TCP Vegas congestion window evolution — FIFO queue
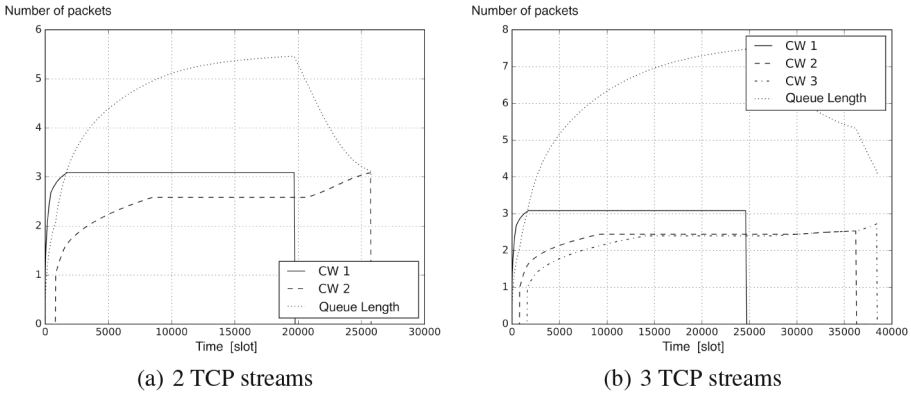
(a) 2 TCP streams   (b) 3 TCP streams

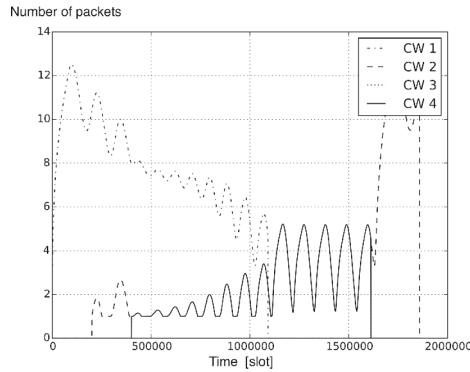**Fig. 2.** TCP Vegas congestion window evolution — RED queue

Fig. 3. TCP NewReno congestion window evolution — FIFO queue, 4 TCP streams

Figure 2 refers to the case of RED queue with two and three TCP streams. Streams start transmission at different time points and TCP Vegas is able to provide a level of fairness much greater than TCP NewReno. Indeed, in such case, as highlighted in Fig. 3, the first stream (starting the transmission with empty links) decreases CWND much slower and uses most of the available bandwidth.

The last set of simulations deals with the friendliness between TCP Vegas and NewReno, considering two connections with the same amount of data to be transmitted. In case of FIFO queue (see Fig. 4(a)), TCP NewReno is more aggressive and sends data faster. Uneven bandwidth usage by TCP variants decreases in presence of the AQM mechanism, as pointed out by Fig. 4(b). Our results confirm that the RED mechanism improves fairness in access to the link and keeps short the queues in routers (in our example, the maximum queue length decreases from 20 to 12 packets).
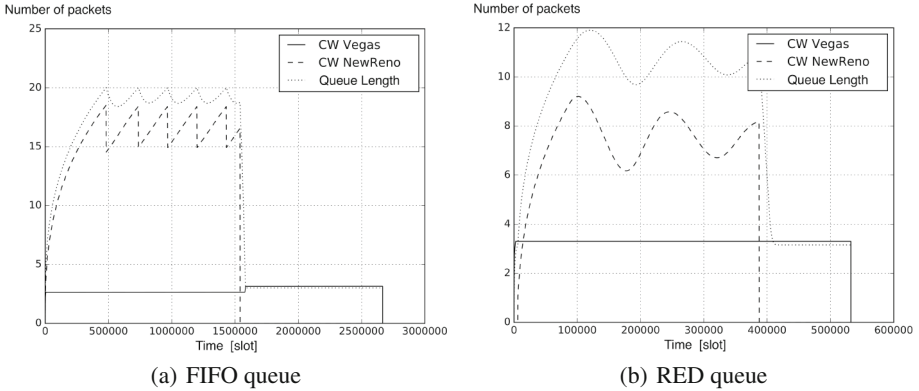
(a) FIFO queue   (b) RED queue

Fig. 4. TCP Vegas and NewReno congestion window evolution

4 Conclusions

The article evaluates by means of a fluid approximation the effectiveness of the congestion control of TCP NewReno and TCP Vegas.

The two TCP variants differ significantly in managing the available bandwidth. On one hand, TCP NewReno increases CWND to reach the maximum available bandwidth and eventually decreases it when congestion appears. This greedy approach clearly favors a stream which starts transmission when the link is empty. On the other hand, TCP Vegas increases CWND only up to a certain level to avoid the possibility of overloading. The disadvantage of this solution is the link underutilization: with a single stream TCP Vegas is conservative and may not use the total available bandwidth. However, in case of several competing streams, TCP Vegas mechanism shows its fairness: in presence of synchronous flows every stream uses the same share of the available bandwidth and even in case of streams starting transmission at different times a quite fair share of the network resources is still obtained.

Finally, the presented analysis permits to take into account finite-size flows and, unlike most works in this area, allows to start TCP transmission at any point of time with different values of the initial CWND (modern TCP implementation often starts with a window bigger than 1 packet). In other words, our approach makes possible the observation of TCP dynamics at such time when other sources start or end transmission.

References

1. Ahn, J.S., Danzig, P.B., Liu, Z., Yan, L.: Experience with TCP Vegas: emulation and experiment. In: ACM SIGCOMM Conference-SIGCOMM (1995)
2. Bonald, T.: Comparison of TCP Reno and TCP Vegas via fluid approximation. Institut National de Recherche en Informatique et en Automatique **1**(RR 3563), 1–34 (1998)
3. Brakmo, L.S., Peterson, L.: TCP Vegas: end to end congestion avoidance on a global internet. IEEE J. Sel. Areas Commun. **13**(8), 1465–1480 (1995)
4. Domańska, J., Domański, A., Czachórski, T., Klamka, J.: Fluid flow approximation of time-limited TCP/UDP/XCP streams. Bull. Pol. Acad. Sci. Tech. Sci. **62**(2), 217–225 (2014)
5. Domański, A., Domańska, J., Czachórski, T.: Comparison of AQM control systems with the use of fluid flow approximation. In: Kwiecień, A., Gaj, P., Stera, P. (eds.) CN 2012. CCIS, vol. 291, pp. 82–90. Springer, Heidelberg (2012). doi:10.1007/978-3-642-31217-5_9
6. Grieco, L., Mascolo, S.: Performance evaluation and comparison of Westwood+, New Reno, and Vegas TCP congestion control. ACM SIGCOMM Comput. Commun. Rev. **34**(2), 25–38 (2004)
7. Hollot, C., Misra, V., Towsley, D.: A control theoretic analysis of RED. In: IEEE/INFOCOM 2001, pp. 1510–1519 (2001)
8. Jacobson, V.: Congestion avoidance and control. In: Proceedings of ACM SIGCOMM 1988, pp. 314–329 (1988)
9. Kiddle, C., Simmonds, R., Williamson, C., Unger, B.: Hybrid packet/fluid flow network simulation. In: 17th Workshop on Parallel and Distributed Simulation, pp. 143–152 (2003)
10. Lee, D., Carpenter, B.E., Brownlee, N.: Observations of UDP to TCP ratio and port numbers. In: Proceedings of the 2010 Fifth International Conference on Internet Monitoring and Protection, ICIMP 2010, pp. 99–104. IEEE Computer Society, Washington, DC (2010)
11. Misra, V., Gong, W., Towsley, D.: Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED. In: Proceedings of ACM/SIGCOMM, pp. 151–160 (2000)
12. Mo, J., La, R., Anantharam, V., Walrand, J.: Analysis and comparison of TCP Reno and Vegas. In: Proceedings of IEEE INFOCOM, pp. 1556–1563 (1999)
13. Wang, L., Li, Z., Chen, Y.P., Xue, K.: Fluid-based stability analysis of mixed TCP and UDP traffic under RED. In: 10th IEEE International Conference on Engineering of Complex Computer Systems, pp. 341–348 (2005)
14. Yung, T.K., Martin, J., Takai, M., Bagrodia, R.: Integration of fluid-based analytical model with packet-level simulation for analysis of computer networks. In: Proceedings of SPIE, pp. 130–143 (2001)