ISSN 2278-3091
Volume 8, No.6, November – December 2019
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse137862019.pdf
https://doi.org/10.30534/ijatcse/2019/137862019

Comparative study of TCP congestion control algorithms

Kaoutar Bazi[1], Bouchaib Nassereddine[2]
[1]Applied Mathematics and Computer Science laboratory, FST, Hassan 1 st University, Settat, Morocco, kaoutar.bazi@uhp.ac.ma
[2]Applied Mathematics and Computer Science laboratory, FST, Hassan 1 st University, Settat, Morocco, nassereddine_bouchaib@yahoo.com

ABSTRACT

TCP (Transmission Control Protocol), is a reliable transport protocol for the transport layer of TCP / IP model, it is the most used transport protocol since it implements panoply of mechanisms ensuring good data transfer. Nowadays, the internet knows a huge growth and therefore it becomes more and more difficult to guarantee the continuity of the services to a very large number of users. This is why a lot of research has been carried out in order to improve the functioning of TCP generally and congestion control more precisely. Several congestion control mechanisms have been proposed to improve the performance of TCP. But, it still suffers from unsatisfactory performances. This is why we will try in this article to conduct a study - based on the analysis of some metrics, packet drop, latency and throughput - for the analysis and comparison of the most powerful of the algorithms proposed in this sense (Tahoe, Reno, New Reno, Vegas, Sack, Fack), in order to identify their advantages and limits under congested environments. So, for the simulation we opted for the NS2 simulator by applying 24 different scenarios for each algorithm. The study showed that TCP Tahoe, Reno, New Reno and Sack are loss-based; they favor the loss of packets to guarantee a short latency. While TCP Vegas is delay-based; it's recommended for applications that require reliable packet transfer. The results of this study will form the basis of future work on the development of a robust algorithm which combines both the advantages of the studied algorithms, and which has the power to share bandwidth fairly with aggressive algorithms such as TCP Reno in order to ensure a good congestion control.

Key words: mesh, congestion control, TCP, IEEE 802.11s, Tahoe, Reno, New Reno, Vegas, Sack, Fack.

1. INTRODUCTION

WMNs 802.11s [1,2,3,4,5,6] are the evolution of the IEEE 802.11 Wireless Local Area Network, it is based on the standard 802.11s, it's an innovative technology that provides miraculous solutions. It can serve a very large number of users by providing them a large bandwidth, low latency with very low costs.A wireless Mesh Network is a multihop wireless network, formed by the association of a certain number of nodes called STA, it is interconnected to the Internet by a set of gateways, and customer traffic is transported using a system multi-hop communication. An 802.11s mesh network device is labeled as a traffic-producing mesh station (STA).The STAs form meshes with each other, on which mesh paths can be established using a routing protocol. The nodes responsible for relaying frames are called Mesh Point (MP), the MP which is an access point is called Mesh Access Point (MAP), the MP can also allows the mesh network to be connected to another external network and in this case it is called Mesh Portal Point (MPP). Internet users demand continuous services with very high performance, and since the WMNs support heavy traffic load, this causes network saturation and subsequently long transfer delays and data loss causing congestion. Indeed, congestion occurs when traffic is generated whose load is greater than the network capacity, the buffers are full and the data is lost.In response to the congestion problem, several congestion management mechanisms have been developed. In this article we are interested in the solutions proposed within the framework of the TCP / IP model, more precisely the techniques aimed at improving the TCP protocol.
TCP (Transmission Control Protocol) is the most used protocol in the transport layer due to its reliability. It has the power to transmit lost packets and in the right order, as it can manage the rate with which it sends and receives segments. Despite all the mechanisms at its disposal, TCP still needs improvements, it is in this sense that the network community has set up several state variables [7] making it possible to diagnose and adapt the state of the network. Among these variables: the acknowledgments (ack) informing the sender that the sent segment has reached successfully the destination, the constant of treshold (ssthresh) controlling the state of the network so that it is not congested, the congestion window (cwnd) managing the data that the network is capable of handling…Therefore, for using these variables, several congestion management techniques have been developed in order to preserve the efficiency of the network and its reliability. The other parts of this paper are organized as follows: The second part is devoted to the description of some TCP variants, then we make a comparative study of the variants already discussed to draw their advantages and limits. And finally, we discuss the results obtained to make conclusions.

2. OVERVIEW AND RELATED WORK

The network community is in continuous quest to put end to the congestion problem. For this, several congestion control

techniques have emerged, deploying all of the network state variables already mentioned. Among others there is :

Additive Increase Multiplicative Decrease (AIMD): Used to prevent congestion, in this technique we start by increasing the rate of sending data little by little as long as there is no congestion, in the presence of the latter we proceed to a sudden decrease. The disadvantage of this technique is that it is not suitable for high-speed networks since it does not allow the use of all of the bandwidth. Slow Start: we start slowly at the beginning of a connection [8], then we increase the rate of sending data rapidly exponentially until the value of the congestion window reaches that of the threshold constant ssthresh. At this point, we wait for a round trip time RTT to increase the congestion window (cwnd) by 1 Maximum Segment Size (MSS). Then there, another technique is brought into play, it is the Congestion Avoidance (CA) [9] where the source sends the segments less quickly in a linear way, and we continue until the reception of duplicated acknowledgments or else the expiration of the Round Trip Time (RTO). At this time we precede other techniques which are Fast Retransmit and Fast Recovery.

All these techniques have been deployed to realize algorithms fighting against congestion. The first algorithm is TCP Tahoe [10], the implementation of this algorithm is based on the use of slow start at the start of a connection until the detection of a loss phenomenon, so here we set ssthresh to the value of cwnd and we reset cwnd to 1 MSS. Thereafter, Congestion Avoidance is followed by a Fast Retransmit upon receipt of a duplicated acknowledgment.So, in TCP Tahoe, we don't wait for the expiration of the RTO. TCP Tahoe could not remedy the congestion problem that is why researchers have thought of improving it by proposing another more efficient version which is TCP Reno [11], In Reno we require the acknowledgment of each segment received. In case of the receipt of several duplicated acknowledgments, we divide the congestion window by a half and we set the constant of threshold to the value of the congestion window, then a phases of Fast Retransmit and Fast Recovery take place. In this way we get a more stable flow than in the case of TCP Tahoe. Since TCP Reno cannot to end the congestion problem, Floyd et al. Thought of improving it by proposing a modified version called TCP New Reno [13, 14]. It's a significant modification to act against the multiple packet losses. For this, they modified the phase of Fast Recovery which is only quit after having acknowledged all the segments contained in the buffer, then cwnd=ssthresh and a new phase of Congestion Avoidance is applied. If there is a lost segment, we send it again and we set the number of duplicated acknowledgement to zero. TCP New Reno wasn't the ideal solution, so Brakmo and Peterson implements a new version of TCP Reno entitled TCP Vegas [12]. The purpose of this version is to achieve a better bit rate by decreasing as possible the lost packets. To do this, they have made certain improvements; decrease the rate of transmission during the phase of slow start, the RTO is verified for each received ack and using a modified congestion avoidance phase. Still in search of the ideal algorithm capable of ending the congestion problem, Mathis and J. Mahdavi proposed a new version called TCP Fack [15], this version is based on the calculation of data which passes through the network and separates control of the congestion from the data recovery. In this way, we guarantee good management of the data flow.

3. PROPOSED WORK

In the previous section we have mentioned some TCP variants while discussing the ways of congestion control used in each with their different techniques. But to better understand the difference between these different flavors, we must experiment them to study their behavior in presence of congestion problems and to distinguish their advantages and their respective limits after having subjected them to the same circumstances.

In this paper our main goal is to study the behavior of TCP variants under congested networks in order to draw conclusions summarizing the strengths and weaknesses of each variant. To do this, we used the NS2 network simulator to simulate a congested network to which we apply one of the TCP variants, each variant undergoes 24 different scenarios depending on the value of the CBR used to weigh down the network (the values of the CBR are varied from 1 to 12 with an increment of 0.5 MB for each scenario).So for configuration, we used a mesh topology whose links between adjacent nodes are 10 MB, all variants are subject to the same conditions and in the same environments. We used scripts to filter and sort the data collected from the simulation, in order to extract the different values (sequence number, flow_id, source, destination, packet_type, packet_size…) which are necessary to have the state values discussed above. . Then, later on, we used these variables to calculate certain metrics essential for the study and evaluation of network circumstances for each variant. The metrics in question are: Throughput, latency and Packet drop rate. To have the different values of these metrics during the simulation, we used the following equations:

$$Throughput = \frac{TotalBitsReceivedByReceiverNode}{FirstPacketSentTime - LastPacketReceivedTime}$$

$$Latency = \frac{\sum (AckTime - SentTime)}{TotalPacketNumber}$$

$$PacketDeliveryRatio = \frac{(TotalSentPacketsNumber - TotalReceivedPacketsNumber)*1}{TotalSentPacketsNumber}$$

4. EXPERIMENT EVALUATION

At this stage, we transform the measurements obtained in the previous section into graphs to facilitate their interpretation. The results obtained are transformed into graphs using the GNUplot tool from NS2. For this we used another script taking as input the values obtained after application of the formulas already mentioned and generating the graphs below, in order to facilitate the analysis of the simulation results and better conduct the comparisons.

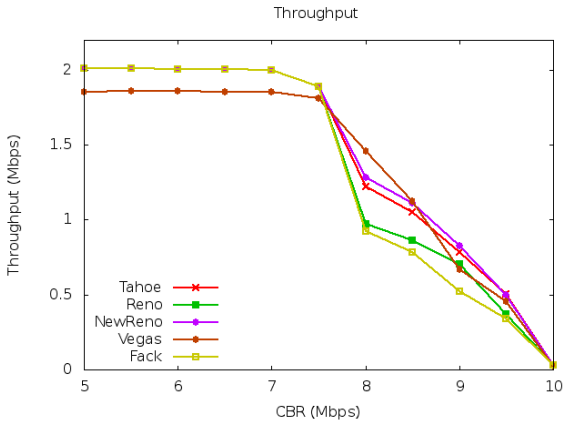

Figure 1: Variation of the throughput according to CBR values

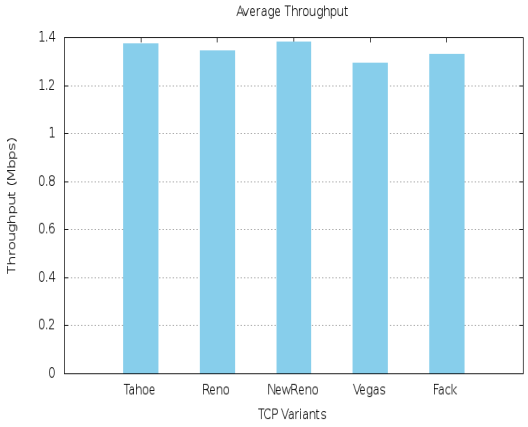

Figure 2: Average throughput of TCP Variants

This graph represents the variation of the Throughput according to the values of the CBR and this for each variant. So, at the beginning of the connection all the variants adopt the same rhythm which is characterized by stability, they even have almost identical values except for TCP Vegas. Subsequently, when the network begins to congestion, the variations change and the values of the Throughput decrease for all TCP flavors, except that TCP New Reno adopts a lighter reduction than the other variants, since it does not wait for the timeout expires to deduct the loss of a packet, and the value of the cwnd is only halved once. On the other hand, TCP Reno knows a remarkable decrease what harms the stability of the flow, especially when it is about a multiple packet losses and it is because cwnd = cwnd / 2 and ssthresh = cwnd. TCP Fack is also experiencing a sudden decrease. For TCP Vegas which is an algorithm which promotes long latency than packet loss, it adopts stable variations with slight decreases. And to consolidate these deductions we have plotted the throughput averages for the five flavors where TCP Vegas has a minimum value while TCP Vegas has the maximum.

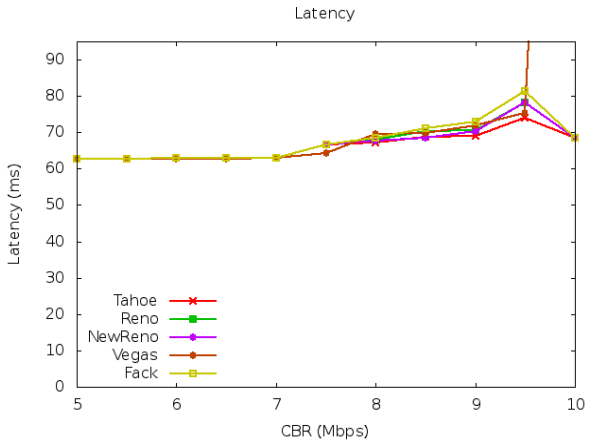

Figure 3: Variation of the latency according to CBR values

In this figure representing the variations of latency, we notice that at the beginning of the connection, the TCP variants have similar values of latency and these values start to differ little by little when the network is congested, this is normal since each variant behaves according to its own strategy and uses its own techniques to deal with congestion. And as already mentioned, TCP Vegas privileges latency it has great values of latency compared to the others. Therefore, it is to be avoided for real time networks.

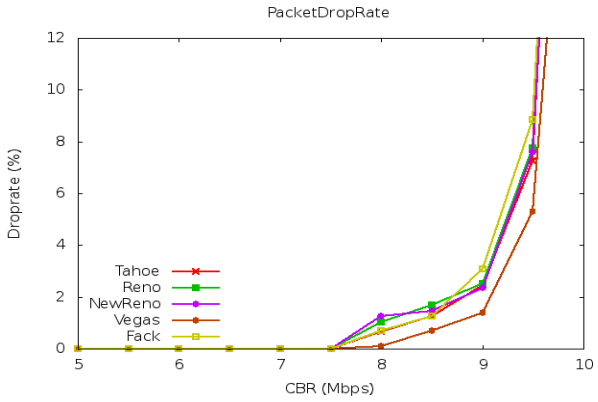

Figure 4: Variation of the drop rate according to CBR values

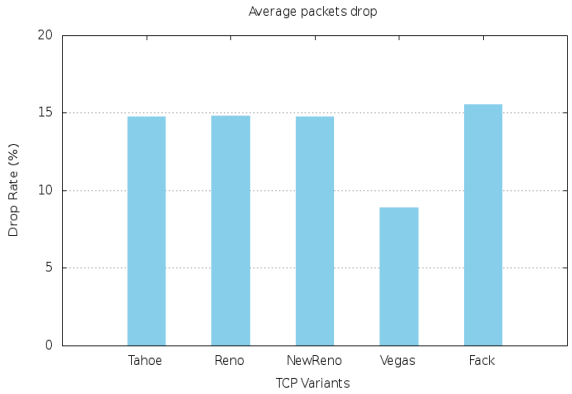

Figure 5: Average packets losses of the TCP Variants

In this figure, the interpretations confirm what we concluded in the previous graphs. as expected, at the beginning there is no packet loss since the network is still lightened. Subsequently, in the presence of congestion, all flavors begin to lose packets with of course minimum values for TCP Vegas which favors latency than data loss as already mentioned. So we can conclude that Vegas is recommended for applications where we don't tolerate the loss of packets. Even in the figure below, we notice that TCP Vegas has the lowest average packet losses.

5. CONCLUSION

In order to put an end to congestion problems, the network community has been trying for quite some time, to find the ideal algorithm capable of guaranteeing good management of the bandwidth between the different concurrent flows, and of providing applications with good throughput even in the presence of congestion. This is why several versions or even improvements of TCP have emerged, among others: TCP Tahoe, TCP Reno, TCP New Reno, TCP Vegas and TCP Fack. According to the simulations carried out, the data collected and the results obtained, we were able to differentiate those which are loss-based (Tahoe, Reno, New Reno and Fack) which are recommended for networks which do not support a long transmission delay, and others that are delay-based such as Vegas. Therefore, it is recommended for networks that require reliable data transfer without loss or corruption. And even for those who are real time, we noticed differences in their reactions to congestion. So we can say that each of the variants discussed has its own characteristics depending on how it deploys the different congestion control techniques, and its own behavior depending on what the state of the network imposes on it. But the major challenge remains how to take advantage of all these variants, their advantages and their limits to give birth to an innovative key solution, capable of adapting quickly to the changes that a network may have;how to take advantage of the available bandwidth in a short time without losing data, how to quickly adapt the transmission rate to network conditions, and how to deal with the problem of fairness between concurrent flows especially in the presence of aggressive algorithms such as TCP Reno.

REFERENCES

1. F. Akyildiz, X. Wang, and W. Wang, ***"Wireless mesh networks: a survey," Computer Networks and ISDN Systems***, 47, 445–487, Mar. 2005. https://doi.org/10.1016/j.comnet.2004.12.001
2. Joseph D. Camp, Edward W. Knightly, 12, ***"The IEEE 802.11s Extended Service Set Mesh Networking Standard", IEE Communications Magazine***, 46, 120-126, August 2008. https://doi.org/10.1109/MCOM.2008.4597114
3. Guido R. Hiertz ; Sebastian Max ; Rui Zhao ; Dee Denteneer; Lars Berlemann, ***"Principles of IEEE 802.11s," in proceedings of 16th International Conference on Computer Communications and Networks (ICCCN), Honolulu, Hawaii***, USA, Aug. 2007. https://ieeexplore.ieee.org/abstract/document/4317949
4. Xudong Wang a, Azman O. Lim, ***"IEEE 802.11s wireless mesh networks: Framework and challenges," Ad Hoc Networks***, 6, 970–984, August 2008. https://www.sciencedirect.com/science/article/abs/pii/S1570870500013 70 https://doi.org/10.1016/j.adhoc.2007.09.003
5. M.Sivaram V.Porkodi (2019). ***FCCP – NS: A Fair Congestion Control Protocol with N-Sinks in Wireless Sensor Networks. International Journal of Advanced Trends in Computer Science and Engineering*** 8(1.2):43-51.
6. Sivaram Murugan, Daroon Hamad (2019). ***Analysis of Mobile IP Wireless Networks in 5G. International Journal of Advanced Trends in Computer Science and Engineering,***
7. Lal Pratap Verma1, Indradeep Verma, Mahesh Kumar, ***"An Adaptive Congestion Control Algorithm," Modelling, Measurement and Control A***, Vol. 92, No. 1, pp. 30-36, March 2019.
8. https://tools.ietf.org/html/rfc5681
9. https://tools.ietf.org/html/rfc2581
10. Biplab Sikdar, Shivkumar Kalyanaraman and Kenneth S. Vastola, ***"Analytic Models for the Latency and Steady-State Throughput of TCP Tahoe, Reno, and SACK", IEEE/ACM TRANSACTIONS ON NETWORKING***, 11, 959- 971, January 2004.
11. Fahad Khan, ***"A Comparative Analysis of TCP Tahoe, Reno, NewReno, SACK and Vegas"***, www.academia.edu/5989428
12. Brakmo, L., Peterson, L. (1995). ***TCP Vegas: end to end congestion avoidance on a global Internet. IEEE J. Sel. Areas Communication***, 13(8): 1465-1480. https://doi.org/10.1109/49.464716
13. Floyd, S., Henderson, T. (1999). RFC2582-***the NewReno modification to TCP's fast recovery algorithm. Internet***

Engineering Task Force. From https://tools.ietf.org/html/rfc2582.

14. Floyd, S., Henderson, T., Gurtov, A. (2004). RFC3782 ***the NewReno modification to TCP's fast recovery algorithm. Internet Engineering Task Force***. From http://tools.ietf.org/html/rfc3782
15. Mathis, M., Mahdavi, J. (1996). ***Forward acknowledgement: Refining TCP congestion control. SIGCOMM Computer Communications Review***, 26(4): 281-291. https://doi.org/10.1145/248157.248181.
16. Tuntun_Oo (2019). ***Design and Implementation of Bandwidth Monitoring, Line Aggregation of VoIP***. International Journal of Advanced Trends in Computer Science and Engineering.