Analyzing Reliable Transport Layer Protocol Performance

Deepika Singh Kushwah¹, Mahesh Kumar² and Lal Pratap Verma³
¹ Research Scholar, CSE Department, Jaypee University of Engineering & Technology, Guna, India
² Associate Professor, CSE Department, Jaypee University of Engineering & Technology, Guna, India
³ Associate Professor, CSE Department, Moradabad Institute of Technology, Moradabad, India
E-mail: ¹deepika.kush09@gmail.com, ²mahesh.chahar@gmail.com*, ³er.lpverma1986@gmail.com

Abstract. Transmission Control Protocol (TCP) is a reliable protocol used for transferring data end to end in the network. It is used by many internet applications. It was originally designed to handle congestion issue in the network. It has many variants like Tahoe, Reno, New Reno. This paper presents the performance of different variants of TCP to identify how they work with varying network characteristics. The simulation is done with the help of NS2. The results show how different variants give their performance on delay, packet drops, and throughput. This is immensely important analysis, to know which variant is better for a specific criterion. This paper covers many variants and their nature of handling congestion regarding to changes in the network.

Keywords: Congestion window, RTT, SRTT, ssthresh, TCP, NS2.

1. Introduction

TCP (Transmission Control Protocol) is a protocol that is used for exchanging data between two end devices in a network. It was introduced back in 1973 and another long eight years to get standardized and RFC793 [1] published after that. Since the introduction of RFC many changes and improvements have been done to TCP. Since TCP is a reliable protocol as it detects loss of data & automatically resends the lost data. It has been stated too that TCP is very powerful in dealing with congestion control & retransmission. Back in 1988 Jacobson [2] discover congestion and his paper gave new insights into congestion control. When the sending rate increases than the receiving rate, the congestion occurs. Some suggestions to deal with congestion were also given. Congestion is the main cause of packet loss in wired networks.

Basically, TCP uses round trip time for estimating its window growth policy to give reliable service in the unreliable network. The estimate of growth of the window is calculated by a simple equation 1.1 given as-

\[ srtt = \alpha \times srtt \times (1-\alpha) \times rt \]  

(1.1)

Also in how much time an acknowledgment should reach is calculated by equation 1.2-

\[ \text{timeout} = srtt + 2\times \text{variance} \]  

(1.2)

With the above equation, the timeout is calculated & if the acknowledgment is not received within this time, the segment is retransmitted.

TCP maintains a congestion the window growth policy to avoid congestion [3,4]. This window indicates the maximum amount of data that can be transferred without being getting lost. If TCP fails to get acknowledgement for some segments within a time period which concludes that congestion has...
been detected. Therefore to deal with congestion, the congestion window should be decreased by half of its current size & then the retransmission is done. TCP variants’ performance is mainly affected by the way they handle congestion window after the congestion is encountered as the amount of data to be transferred depends on the congestion window. Some contributions provide less congestion prone environment for smart cities (in case of wireless networks) to help networks deploy easily on different locations & centerline huge data amount from different parts of the cities [4].

Many variations have been done in this congestion detection and the way to handle congestion [5]. These can be stated as variants of TCP or different flavors of TCP. Some authors have done the congestion control for heterogeneous internet too [6]. On the other hand, comparison of different flavors of TCP has also done in MANET and their behavior has been analyzed [7]. However many TCP variants have been explained in this paper. Also how these variants behave differently when the rate of data transmission varies. The simulations results are also shown in another section which gives a clear vision of the behaviour of these variants.

2. Literature Survey

In [8] authors discuss how the TCP behaves with specifications of network i.e. bandwidth, delay, time to live, round trip time, sending rate of packets, etc. and conclude which particular variant behave best on which parameter. Hence an appropriate version is selected according to the specification. In another paper [9] author evaluates the performance of variants like Tahoe, Reno, New Reno, SACK, FACK & Vegas on a simulator NS2. After comparing all the variants of TCP, the authors concluded that Vegas perform better than all other variants to send data with the least chances of occurrence of congestion. In another study [10] authors did a comparative study on different parameters of five TCP variants which easily let us know which characteristic is supported by which variant. In [11], the authors compare three variants i.e. Reno, New Reno & full TCP on different routing approaches namely AODV, DSR & DSDV in Grid Topology. The results given by NS2 shows that Full TCP performs better than the other two approaches without getting affected by the routing approach. In [12], the author took three network parameters namely throughput, packet drop ratio & latency & three different types of experiments done by them. In the first one, variance, mean & T-test analysis done & Vegas perform better in this experiment. In the second one, if there are two flows & both using the same variant, they behave fairly to each other in bandwidth utilization but when different variants are used, they become unfair. The third experiment illustrates that using different queuing algorithm i.e. drop-tail & RED both gives better results in terms of throughput & delay respectively. In [13] the, author compares different variants of TCP with different parameters of AODV with NS2 which results that Vegas gives higher efficiency of all. In another paper [14], the delayed acknowledgments also have some impact on the performance of TCP. Results indicate that Vegas achieves better performance than New Reno.

3. TCP Variants:

3.1. Tahoe:

Initially, TCP used the ‘Go Back N’ approach to control network congestion. Then TCP Tahoe came into existence. Tahoe added slow start, congestion avoidance & fast re-transmit approach to handle congestion in the network. It follows the conservation policy of packets i.e. if a connection is having some bandwidth capacity, the packet will be injected only when the other packet is taken out. For this purpose, acknowledgments are used. For implementing this policy, a congestion window is maintained which shows the network capacity [14].

Tahoe contains these phases- (i) Slow Start, (ii) Congestion Avoidance

(i) Slow Start- Data transmission starts when the window size is one. After receiving the acknowledgment of this one packet, the window size is incremented exponentially. Therefore the second time it sends two packets & so on the process continues. Also, this can be stated as cwnd+ = MSS.
(ii) Congestion Avoidance- in this phase Tahoe uses Additive Increase Multiplicative Decrease. When a packet is lost, it is taken as an indication of the congestion occurrence. Then the congestion window is halved from the current size of congestion window as the threshold value. Again the congestion window is started from one to threshold value. A timeout is required to deal with packet loss as it uses cumulative acknowledgments.

The issue with Tahoe is that it doesn’t utilize the bandwidth well. During the course of timeouts, most of the time, bandwidth of the channel is wasted & more time is required for getting the lost packet again.

3.2. Reno:
It includes algorithms called ‘Fast Retransmit’ & ‘Fast Recovery’ for congestion control. It is the successor of Tahoe with all the functions of Tahoe & the additional two functions mentioned previously in the section. So, ‘Fast Retransmit’ activates when it receives three duplicate acknowledgments. These three duplicate acknowledgments conclude that the segment previously sent has been lost [15]. Therefore the lost segment needs to retransmit immediately and it enters in ‘Fast Recovery’ mode.

- Slow Start threshold is set to half of the current window size & we got a new congestion window now.
- When receiving duplicate acknowledgments, the window is incremented by one each time it sends a packet successfully. If the window is increased greater than the amount of data in the pipe, then a new segment is transmitted or it waits otherwise.
- After retransmitting a segment, one RTT wait should be there so that a fresh acknowledgment gets some time to be received. This reduces the pipe to be empty but reduces the flow.
- In the fast recovery phase, the new congestion window (CWND) is set as CWND= 1/ Current CWND.

When packet loss is small, Reno performs well. But, if there are multiple losses then Reno does not give good performance and behaves same as Tahoe [15].

Another issue arises when the window is very small. Because when a packet loss occurs, there are not sufficient packets to generate duplicate acknowledgement thrice, needed to enter in the fast retransmit phase. The only solution left is the timeout.

3.3. New Reno:
This is the modification to TCP Reno. It can detect multiple losses events. Also likewise Reno, when loss (single or multiple) occurs, it enters into the fast retransmission phase. The difference between Reno & New Reno is that until the outstanding data is fully acknowledged, New Reno remains in the Fast Retransmission phase [16]. Working of the Fast Retransmission phase in Reno & New Reno is the same. Fast Recovery behaves differently in both variants of TCP. New Reno introduces a term called Partial Acknowledgment. This partial acknowledgment is a slight modification done in Fast Recovery mode acknowledgement. In partial acknowledgment, it acknowledges data that is new but does not acknowledges the overall outstanding data left in the pipe when loss detects. The issue with this variant is to detect packet loss, one RTT is needed. Only when one segment acknowledgment is received then the information of the other segment lost is concluded.

3.4. SACK (Selective Acknowledgement):
It is also the extension of TCP Reno & that uses the SACK option proposed by IETF. It deals with multiple packet losses. It retransmits more than one packet loss per RTT. SACK tackles the issue of multiple packet drops from a single window of data. Also, when the aforementioned condition occurs, the TCP has to wait for the retransmission timer expiration & after that only, the data flow is initiated. Without SACK both the previous TCP variants, i.e. Reno & New Reno senders retransmit at most one dropped the packet in a single RTT. Also, SACK gives a clear idea of packets delivered successfully or not. Therefore, unnecessary retransmission & delay can be avoided & as a result, the performance of TCP is improved with this flavor. The main difference between TCP Reno & TCP SACK is the
way of dealing with multiple packet loss within a single window. SACK treats fast recovery differently when the sender receives duplicate acknowledgments. Then the sender retransmits a packet & cuts the window in half. Also, it maintains a variable that shows the number of outstanding packets in the path [17].

SACK behaves the same as Reno on a single loss. Also follows conservation of packets policy on fast recovery mode. SACK doesn’t behave well with some network conditions [17] i.e. when it shares paths with Reno. The sender only sends new or retransmitted data when an old packet gets out of the pipe. Depending on the variable, it is decided when to send the packet & which packet to send. Also, if the transmitted packet is dropped only, SACK detects the drop after a retransmission timeout occurs.

3.5. VEGAS:
It is a modification done on Reno. Also, VEGAS uses a proactive approach as a comparison with other flavors of TCP that uses a reactive approach. VEGAS does not wait for the occurrence of congestion but rather detects congestion & try to avoid it. VEGAS uses a system clock & go for the calculation of RTT & records the timestamp for the relevant segment. The approaches of VEGAS are different in the case of retransmission & congestion avoidance.

- In the retransmission, VEGAS tracks each segment sending time & estimates RTT.
- On receiving a duplicate acknowledge it checks the difference of current time with that of the segment’s transmitted time, if the latter is greater than estimated RTT.

If the difference is positive, VEGAS immediately transfers the segments without waiting for a third duplicate acknowledgment. Some studies [18] say that VEGAS retransmits one-fifth to one-half as much as data as Reno. VEGAS detects congestion or losses much sooner than Reno.

The disadvantage of Reno is overcome by VEGAS as it isn’t able to detect loss when the window is small enough for duplicate acknowledgments. Another is it detects multiple packet losses.

If the retransmitted segment was sent after the window decreased then only the window is reduced. Therefore the issue of multiple times window reduction of Reno is overcome by VEGAS.

Congestion Avoidance in VEGAS: as prior mentioned it proactively deals with congestion. Detection of congestion is done in such a way that it checks if there is any decrease in sending rate with the expected rate. So if the queue starts forming, the sending rate will ultimately drop. Therefore if the difference between both rates becomes less, VEGAS decreases its transmission rate. Because of this treatment of transmission rate, bandwidth wastage is also controlled by VEGAS. Therefore VEGAS performs better than SACK too, as it efficiently estimates congestion along with throughput. Another drawback of SACK is overcome by VEGAS i.e. it is more stable than the prior as it keeps track of sending rate & keeps on incrementing it unless congestion detected.

3.6. Westwood:
It is a modification of Reno which improves the performance in wired as well as wireless networks. The recognizable differences can be seen in wireless networks that have lossy links.

Working of TCPW: it measures the bandwidth at the sender’s side continuously with the help of acknowledgments returning rate. With this episode, congestion detection is done. Therefore, the congestion detection is done [19]. Therefore, congestion window, slow start threshold is computed after getting three duplicate acknowledgments or timeout.

The difference with Reno is Reno halves the congestion window aggressively after three duplicate acknowledgments. Despite halving the window, Westwood selects a ssthresh with consideration of bandwidth used at the occurrence of congestion & further named this approach as “Faster Recovery”. Further, the experiments done by using the aforementioned approach give improved throughput & in fairness too. Also, when working with Reno, the performance of Reno isn’t deteriorate by TCPW.

3.7. Hybla:
To deal with issues like high RTT & high link error of satellite network Hybla came into the picture. Due to these factors, congestion window of traditional TCP protocol doesn’t utilize the available
bandwidth. Hybla solves the aforementioned issues of satellite networks which are also known as heterogeneous i.e. terrestrial and satellite networks.

First of all, Hybla removes the dependency on RTT to measure the performance of the network which is done by many previous TCP flavors [20]. Usually, window is triggered by the arrival of acknowledgments. If the RTTs are long, it results in the reduction of window & throughput is degraded too. Due to this bandwidth is shared unfairly. TCP Hybla solves the issue of inequality in RTT. It mainly focuses on how to prevent connections that have high delays only. It obtains the transmission rate of packets of wired & wireless both networks i.e. for wired it uses the reference of TCP connections & long RTT connections for wireless [21]. Therefore Hybla doesn’t depend on RTT for transmission of data hence it provides maximum throughput. But Hybla makes the performance of the network lesser predictable & as a result many applications don’t use it.

3.8. CUBIC:
We have another variant of TCP & which is currently used by LINUX. Cubic is the advanced version of the TCP-BIC (Binary Increase Congestion Control) algorithm. BIC uses the concave & convex portion of a cubic function for the growth of the window. This variant modifies the linear window growth function that is being used by existed TCP flavors. Due to this, Cubic improves the scalability of TCP in fast & long networks. It also makes the growth function of windows independent of RTT. Therefore the congestion window of all flows grows at the same rate.

The main feature of Cubic is that the window growing policy depends on the real-time between two consecutive congestion events [21]. The congestion epoch is the real-time where one congestion event goes into the fast recovery phase. Therefore the growth of the window doesn’t depend on RTTs. Therefore it aids in the same bottleneck issue. Since window growth rate is fixed when RTTs are short then the growth rate becomes slower than standard TCPs. These overcomes are managed by SACK only. CUBIC does several upgrades in kernel of LINUX.

- It forms a plateau called $W_{\text{max}}$ & if the congestion window is less than $W_{\text{max}}$, after receiving an acknowledgment, then the concave region will be used by the protocol.
- If the window size is larger than $W_{\text{max}}$ then the Cubic is in convex function.

Since the Internet is unpredictable therefore bandwidth fluctuates. In the convex region, the window grows slow in the beginning & increases gradually & finds a new $W_{\text{max}}$. Cubic also improves the convergence speed of the network [21]. Therefore whenever a loss is encountered, it is reflected very fast.

![Figure 1: Window Growth Function of Cubic](image)

3.9. Highspeed:
Standard TCP works well in low-speed communications but became worse at high speed. Therefore some solutions were proposed & Highspeed TCP is one of them. In this, modifications are done on the server side of TCP.
The need for Highspeed TCP arises because sometimes the usual issue in the network arises are when available between is less than the required bandwidth & congestion. Since congestion is a common phenomenon that needs to be solved in the case of a Highspeed network. So Highspeed TCP protocol has been introduced to deal with this aforementioned issue. Many variants were introduced to deal with the issue.

In these approaches, buffer delay is considered as a sign of occurrence of congestion other than the dropping of packets [23]. Some say [23] that ‘X’ channels should be opened virtually but the deciding factor of X is tricky. Now, moving to Highspeed TCP since it is a flavor of TCP therefore compatible with all the previous versions.

Highspeed TCP behaves the same as standard TCP for small window size whereas, in heavy congestion the networks, it doesn’t affect the behavior of network. If concluded, the effects of Highspeed in different conditions: It performs better than standard TCP variants in normal or default conditions. Also when the bandwidth availability is high, it performs better. And in large & small buffer sizes. The performance of highspeed is far better than other variants.

It is easy to implement as changes are required to be done on the server side only. Small changes in the congestion window updating method improve throughput by 32%.

4. Experimental Analysis
We have analyzed the behavior of different variants of TCP with the help of simulator NS2. The scenario on which the simulation is done is: we have taken four nodes with two nodes working as sources & one destination with one intermediate node. Various TCP variants i.e. Reno, New Reno, Vegas, Westwood, CUBIC, Highspeed & Hybla are compared with the same sending data rate. Packet drops in all variants are compared & it is being concluded that maximum packet loss is in CUBIC & minimum packet loss is in Vegas as Vegas is a proactive approach when dealing with congestion. The current congestion control algorithm used by LINUX is CUBIC. The figure below shows the relativeness of the dropping of packets with the changing rate of sending data in various variants of congestion. So when we look for a minimum packet drop ratio, it is suggested to use Vegas.

Also, simulation results show the throughput generated by various variants in the same conditions with varying rates of data transfer. It is being recommended to use CUBIC which gives maximum throughput among other TCP variants. This could be one of many reasons why LINUX is using CUBIC as its current congestion control algorithm. It is being notable that Vegas generates the worst
throughput whereas previous packet drops are minimum in Vegas. Figure 3 shows the comparison of the throughput of various variants of TCP. From figure 4, it can be easily stated that delay is very less in Vegas from various other variants of TCP. One of the reasons could be the proactive approach that Vegas used to detect congestion. The difference in all the variants is way the deal with congestion. All other than Vegas adopt a reactive approach imprecisely.

![Figure 3: Throughput of TCP](image)

![Figure 4: Delay of TCP](image)

Out of many reasons for the delay could be reordering of packets at the receiver side. This reordering sometimes need more time than usual which ultimately causes delay [24]. Sometimes background traffic also adverse the performance of TCP which results in delay, lower throughput, and high packet drops, etc. [25-26].

5. Conclusion
This paper presents a comparison among various variants of TCP. The comparison is done by running a simulation scenario many times in NS2. We have calculated various TCP variants i.e. Tahoe, Reno,
New Reno, Vegas, Westwood, CUBIC, Highspeed, Hybla & HTCP. The metrics used for comparing the performance of various variants are throughput, delay & packet drop. After analyzing performance from simulated results & obtained graphs, we have found that Vegas performs better than any other Variant of TCP when it comes to packet drop hence it can be used for sending data & information. But on one hand, it least drops packet & also has given lesser throughput in some situation. This is because of the proactive approach of Vegas used to detect congestion whereas variants other than this use reactive approach i.e. they increase their window until a loss of a packet is detected. Also, the delay is less in Vegas as compared to other variants. We have also given the details on the behavior of all of these variants previously in the paper.

6. References
[1] Postel, J., “Transmission Control Protocol”, September 1981, RFC 793, Darpa Internet Program.
[2] Jacobson, V. 1988. “Congestion Avoidance and Control.” ACM SIGCOMM Computer Communication Review 18 (4). Association for Computing Machinery (ACM): 314–29. doi:10.1145/52325.52356.
[3] Luan, Gan, and Norman C. Beaulieu. 2020. “Accurate Mathematical Modeling and Solution of TCP Congestion Window Size Distribution.” Computer Communications 163 (November). Elsevier BV: 195–201. doi:10.1016/j.comcom.2020.09.010.
[4] Lin, Jinting, Lin Cui, Yuxiang Zhang, Fung Po Tso, and Quanlong Guan. 2019. “Extensive Evaluation on the Performance and Behaviour of TCP Congestion Control Protocols under Varied Network Scenarios.” Computer Networks 163 (November). Elsevier B.V. doi:10.1016/j.comnet.2019.106872.
[5] Mudassar, Ahmad, Ngadi Md Asri, Ahmad Usman, Kashif Amjad, Ibrahim Ghafir, and Mounir Arioua. 2018. “A New Linux Based TCP Congestion Control Mechanism for Long Distance High Bandwidth Sustainable Smart Cities.” Sustainable Cities and Society 37 (February). Elsevier Ltd: 164–77. doi:10.1016/j.scs.2017.11.005.
[6] Wang, Zhiming, Xiaoping Zeng, Xue Liu, Man Xu, Ya Wen, and Li Chen. 2016. “TCP Congestion Control Algorithm for Heterogeneous Internet.” Journal of Network and Computer Applications 68 (June). Academic Press: 56–64. doi:10.1016/j.jnca.2016.03.018.
[7] Abu-Zant, Mahmoud, and Mohammad Hamarsheh. 2017. “A Comparison of Congestion Control Variants of TCP in Reactive Routing Protocols MANET.” International Journal of Computer Science and Information Technology 9 (6). Academy and Industry Research Collaboration Center (AIRCC): 25–33. doi:10.5121/ijcsit.2017.9602.
[8] Md. Shohidul Islam, M.A Kashem, W.H Sadid, M. A Rahman, M.N Islam, S. Anam 2009. “TCP Variants and Network Parameters: A Comprehensive Performance Analysis.” 2009. Proceedings of the International MultiConference of Engineers and Computer Scientists 2009 Vol I IMECS 2009, March 18 - 20, 2009, Hong Kong
[9] B N, Prof. Yuvaraju, and Dr. Niranjnan N Chiplunkar. 2010. “Scenario Based Performance Analysis of TCP Using NS2-Simulator.” International Journal of Computer Applications 4 (9). Foundation of Computer Science: 20–24. doi:10.5120/855-1197.
[10] Kaur, Harjinder, and Gurpreet Singh. 2017. “TCP Congestion Control and Its Variants.” Vol. 10. http://www.ripublication.com.
[11] Das, Namita, Sukant Kishoro Bisoy, and Sanjukta Tanty. 2019. “Performance Analysis of Tcp Variants Using Routing Protocols of Manet in Grid Topology.” In Advances in Intelligent Systems and Computing, 768:239–45. Springer Verlag. doi:10.1007/978-981-13-0617-4_23.
[12] Sawarkar, A. and Saraswat, H., 2016, “Performance Analysis of TCP Variants”, International Journal of Computer Science and Network Security, vol. 16, pp. 102-106.
[13] Neha Bathla , Amanpreet Kaur & Gurpreet Singh “Estimating Performance of TCP Alternatives in Wireless Environment.” National Conference on Current Research Trends in Cloud Computing & Big Data (CRTCBB-2015) 81-89, 2015,Jaipur DOI: 10.13140/RG.2.1.1637.4880.
[14] Fall, Kevin, and Sally Floyd. 1996. “Simulation-Based Comparisons of Tahoe, Reno, and SACK TCP.” *Computer Communication Review* 26 (3). Association for Computing Machinery (ACM): 5–21. doi:10.1145/235160.235162.

[15] Miyani, Hardik V, Vishv B Kukadiya, M R Kapil, S Raviya, and M R Dhrumil Sheth. n.d. *Journal Of Information, Knowledge And Research In Electronics And Communication* “Performance Based Comparison Of Tcp Variants ‘Tahoe, Reno, Newreno, Sack’ In Ns2 Using Linux Platform.”

[16] Moraru, Bogdan, Flavius Copaciu, Gabriel Lazar, and Virgil Dobrota. n.d. “Practical Analysis of TCP Implementations: Tahoe, Reno, NewReno.”

[17] Floyd, Sally. 1996. “Issues of TCP with SACK.” *Time*, 1–5. http://en.scientificcommons.org/43064927

[18] Brakmo, Lawrence S., and Larry L. Peterson. 1995. “TCP Vegas: End to End Congestion Avoidance on a Global Internet.” *IEEE Journal on Selected Areas in Communications* 13 (8): 1465–80. doi:10.1109/49.464716.

[19] Casetti, Claudio, Mario Gerla, Saverio Mascolo, M. Y. Sanadidi, and Ren Wang. 2002. “TCP Westwood: End-to-End Congestion Control for Wired/Wireless Networks.” *Wireless Networks* 8 (5): 467–79. doi:10.1023/A:1016590112381.

[20] Caini, Carlo, and Rosario Firrincieli. 2004. “TCP Hybla: A TCP Enhancement for Heterogeneous Networks.” In *International Journal of Satellite Communications and Networking*, 22:547–66. doi:10.1002/sat.799.

[21] Trivedi, Siddharth, Sanjay Jaiswal, Rituraj Kumar, and Shrish Rao. 2010. “Comparative Performance Evaluation of TCP Hybla and TCP Cubic for Satellite Communication under Low Error Conditions.” In 2010 *IEEE 4th International Conference on Internet Multimedia Services Architecture and Application, IMSAA 2010*. doi:10.1109/IMSAA.2010.5729424.

[22] Ward, Morgan. 1959. “The Vanishing of the Homogeneous Product Sum of the Roots of a Cubic.” *Duke Mathematical Journal* 26 (4): 553–62. doi:10.1215/S0012-7094-59-02652-3.

[23] Dalton, Lori A., and Ciji Isen. 2004. “A Study on High Speed TCP Protocols.” In *GLOBECOM - IEEE Global Telecommunications Conference*, 2:851–55. doi:10.1109/glocom.2004.1378080.

[24] Feng, Jie, Zhipeng Ouyang, Lisong Xu, and Byrav Ramamurthy. 2009. “Packet Reordering in High-Speed Networks and Its Impact on High-Speed TCP Variants.” *Computer Communications* 32 (1): 62–68. doi:10.1016/j.comcom.2008.09.022.

[25] Ha, Sangtae, Long Le, Injong Rhee, and Lisong Xu. 2007. “Impact of Background Traffic on Performance of High-Speed TCP Variant Protocols.” *Computer Networks* 51 (7): 1748–62. doi:10.1016/j.comnet.2006.11.005.

[26] Callegari, C., S. Giordano, M. Pagano, and T. Pepe. 2012. “Behavior Analysis of TCP islam Variants.” *Computer Networks* 56 (1): 462–76. doi:10.1016/j.comnet.2011.10.002.