International Journal of Hybrid Information Technology
Vol.6, No.5 (2013), pp.297-332
http://dx.doi.org/10.14257/ijhit.2013.6.5.28

Novel Slow Start Algorithm

Ivan Petrov[1] and Toni Janevski[2]

[1]*Makedonski Telekom, Orce Nikolov bb, 1000 Skopje, Macedonia,*
[2]*Ss. Cyril and Methodius University in Skopje, Faculty of Electrical Engineering and Information Technologies, Karpos 2 bb, 1000 Skopje, Macedonia,*

Ivan.Petrov@telekom.mk, tonij@feit.ukim.edu.mk

Abstract

Internet has developed into dynamic network where users use wired/wireless access technologies. The diversity of bandwidths, delays and error rates observed on Internet have increased. TCP is the common denominator for many services therefore by modifying TCP the need for applying solutions locally can be reduced. We focus on making TCP more efficient in a way to speed up its slow start phase. Different Slow Start algorithms will be analyzed and new variant will be proposed.

Keywords: *Slow Start; cwnd; TCP; congestion; throughput*

1. Introduction

Internet provides best effort service, the network does it's best to deliver the data as efficiency as possible. File downloading today could take twice the time it took yesterday. In the 90 ties more than 90% of the Internet traffic was TCP based. The base shape of TCP algorithm was presented by Van Jacobson [1-2]. Exponential increase of the sending rate was introduced; the initial window size was set at three packets. This protocol was offering fast data transfer, reliable connection, congestion control and flow control. The mechanism was efficient in time when 56K modem communication was a standard. Today the broadband links are defined as connections that provide speed faster than 1Mbps. With the time TCP was evolving, variety of protocols were announced, each of them designed to provide better network efficiency. Slow Start, congestion avoidance and flow control are the main phases in which they differ. TCP was developed to provide best wired network utilization. Limited Slow Start [3], Larger Initial Window, TCP Reno, New Reno, Vegas, Sack, Dual, Fast, Bic, Cubic, Highspeed, Hamilton, Hybla, Scalable, Westwood, Veno, Low priority, Illinois, Compound, Snoop, Abs, Fack, Linux, Full are some of the algorithms and TCP versions that were developed to improve the network utilization [4-18]. Most of the developed protocols were targeting wired network utilization; work was done in wireless traffic utilization too. Initial network conditions are changed; today we are dealing with broadband networks, the need of faster slow start phase is obvious if we want to provide better network utilization. This is the main driver of our research and motivates us to modify the slow start phase. The paper is organized as follows: Section 2 gives brief overview of the slow start phase, discusses some related work and motivates the need of our approach. Section 3 describes our simulation environment and section 4 presents the simulation results. Section 5 concludes the paper.

ISSN: 1738-9968 IJHIT

2. TCP Evolution

Early TCP versions included a method for the receiver to control the rate at which the sender was transmitting but no algorithms for handling dynamic network congestion which was the main reason why the networks suffered from congestion collapses. Numbers of algorithms were proposed by Jacobson which were incorporated in the Tahoe version. The main idea was the TCP sender continuously to adapt its sending rate to the available network capacity. This TCP regulates its sending rate by maintaining a set of windows, sending window (swnd), congestion window (cwnd), and the receivers advertised window (rwnd) where swnd is the minimum of cwnd and rwnd and it determines the sending rate. To find an appropriate value for cwnd, TCP continuously probes for available bandwidth by increasing the sending rate. How fast TCP will increase the sending rate is predefined by the phase in which it is (slow start or congestion avoidance). In slow start, cwnd is doubled each rtt leading to an exponential increase of the sending rate. When the sender is close to the estimate of the network capacity it is usually in congestion avoidance phase and increases cwnd by one segment per rtt. Slow start threshold (ssthresh) determines in which phase the sender is in (slow start or congestion avoidance). In case of loss ssthresh is set to max (Flight_size/2,2*SMSS), this creates history of the network capacity. Flight_size is the amount of data that have been sent but not yet acknowledged and has similar value to cwnd. SMSS defines the size of the largest segment that the sender can transmit. Slow Start is used if cwnd is less than ssthresh and optionally if cwnd equals ssthresh. TCP variants vary in the way they approach to solve the losses and network congestion. TCP Reno for example reduces the sending rate to half the prior sending rate when a loss is detected through the receipt of dupthresh dupack. This is called fast recovery phase, this phase provides Reno to provide higher sending rate than Tahoe. Fast retransmission is common phase for the both protocols. TCP probing behavior is presented at Figure 1. At the beginning the sender is in the slow start. When cwnd exceeds ssthresh the sender enters the congestion avoidance phase. After detecting loss through the arrival of dupthresh dupacks, cwnd is set to one for Tahoe and to half the prior cwnd in case of TCP Reno. Cumulative acknowledgements can inform the sender of one missing segment at time. In networks with long delays and high bandwidths this is a problem, it means that the sender either has to retransmit all segments starting from the lowest unacknowledged byte or should wait an rtt after each retransmission to get an indication of any other missing segments. This causes frequent timeouts if more than one segment is lost during the same rtt. New Reno was introduced it was designed to avoid multiple fast retransmit periods. TCP SACK was introduced to provide retransmission optimization and timeout prevention. It uses the option field. The length of this field is variable, but is limited at 40 bytes. SACK option allows the boundaries of at most four contiguous and isolated blocks of received segments to be specified. This option was quickly implemented in the systems and was used to estimate the amount of data that is currently in the network, which determines when the segments can be sent. The mentioned algorithms fall into the AIMD (Adaptive Increase, Multiplicative Decrease) family of algorithms. TCP addresses three major issues: reliability, flow control and congestion control. Congestion control is achieved by adapting of the sending rate. TCP feedback for successfully delivery of a packet is embodied by returning acknowledgement (ACK). Competing TCP senders with different end to end propagation delays will typically receive feedbacks at different rates and adapt their sending rate at a different price. A number of protocols were designed to limit the effects of the RTT unfairness and to improve the scalability over gigabit links. Most of them adopt a proactive approach based on monitoring packet's RTT and reacting to its increase in an attempt to avoid network congestion. This behavior is justified by the assumption of a strong correlation between packet loss and RTT increase prior to the loss event. Examples of

algorithms that fit into this category are TCP Vegas, TCP Dual, Fast TCP. TCP Hybla implements a constant increase algorithm and provides RTT fairness under a certain stability bound. TCP Cubic tries to decouple the window growth from the returning ACK's that is similar with the approach proposed by H-TCP. With Cubic the window size is a function of the time elapsed since the last packet loss, thus allowing higher efficiency in terms of total bandwidth utilization in case of long RTTs and reducing the throughput dependency from the RTT. In our scope of observation will be the AIMD based algorithms. In the previous work [19] we have found that best throughput performances were achieved when TCP Sack was used as transport protocol this is the main reason why we will use it in the simulation. If we modify the TCP Slow start phase in a way to achieve faster bandwidth utilization than TCP will be optimized to provide better throughput in broadband networks. If we take in consideration the time and conditions under which the protocol was born than from this point of view the improvement of the Slow start phase is important.

2.1 Slow Start

TCP uses a window-based congestion control. In the beginning nothing is known about the network. To get a good starting value for the congestion control, TCP uses an algorithm called Slow-Start [1]. Slow-Start starts with a congestion window (cwnd) of 1 (or something larger, when using larger initial window). As soon as the first TCP packet is ACK ed (the corresponding ACK packet has arrived) cwnd will be increased by 1. Now 2 packets will be sent. For each ACKed packet cwnd will be increased by 1. This behavior doubles cwnd for each RTT. The size of cwnd can be expressed by the following formula (t is the time since the beginning of the connection, it is a multiple of RTT):

$$cwnd = 2^{t/RTT} \qquad (1)$$

The formula shows that cwnd increases exponentially. It is a good approximation as long as the network is able to transmit the whole traffic burst before the first packet of the next burst reaches the first queue that contains segments of the previous burst. We can conclude that Slow start uses an exponential function which increases slowly for small values and quickly for larger values. This means that several RTTs will be needed to achieve the bandwidth but two time's overshoot of the network capacity is also possible. In order to combine the Slow Start and the AIMD algorithm Slow start threshold variable was introduced. Limited Slow start was proposed [3] because of the aggressive behavior of the standard Slow Start for large congestion windows. New parameter was introduced with predefined value of 100 MSS. If cwnd has value smaller than 100 MSS than standard Slow Start algorithm is used otherwise LSS is activated. LSS increases the cwnd value slower, this increases the time needed to reach the bandwidth but on the other hand in case of overshooting the network the losses will be smaller.

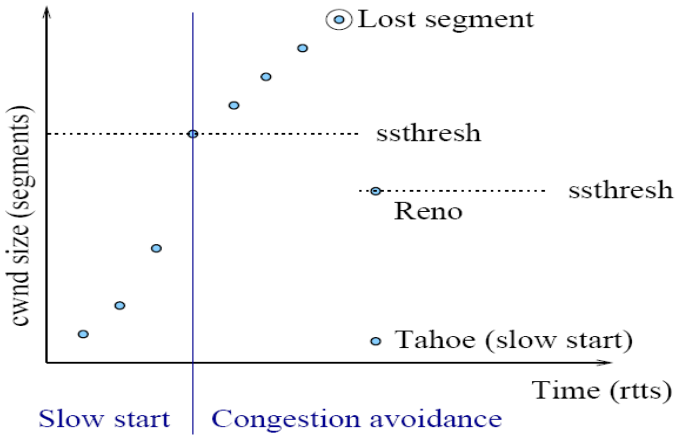

Figure 1. TCP behavior (Slow start and Congestion avoidance phase)

Other method to speed up the Slow Start phase is to use larger initial window, this mechanism will speed up the beginning of the connection but the worst case will be overshoot of twice the available bandwidth. In our simulation scenario we are going to use TCP Sack as a transport protocol with slow start that uses faster increasing function. Like we state before faster increasing function adapts faster at the available bandwidth but it can overshoot the available bandwidth several times. In the basic Slow start it is twice the available bandwidth. LSS will be incorporated because of its function to limit the number of dropped packets for a large cwnd.

2.2 Proposal of Advanced Slow Start mechanism (ASS)

The advanced version of Slow start is proposed in the following algorithm:

Algorithm 2
if cwnd < 25: cwnd+=3;
if cwnd >=25 && cwnd<50: cwnd+=2
if cwnd >=50 && cwnd<75: cwnd+=1
if cwnd >=75 && cwnd<100: cwnd+=0.5
if cwnd >=100 use LLS

The decrease parameters of cwnd in case of loss are chosen in a way to minimize the network overshooting. Algorithm 3 and 4 are more aggressive versions of the second algorithm. They will be used as guideline and comparative mechanisms that will provide results to indicate if we are not at the right track.

Algorithm 3
if cwnd < 25: cwnd+=2;
if cwnd >=25 && cwnd<100: cwnd+=1
if cwnd >=100 use LLS

Algorithm 4
if cwnd < 50: cwnd+=2;
if cwnd >=50 && cwnd<100: cwnd+=1
if cwnd >=100 use LLS

The choice of the threshold values can be explained with the usage of the 100 by LSS as special number. LSS round function changes every 50 packets so if we decrease 100 for 50 packets we well have the threshold of the third algorithm. 25 is the consequence of the LSS algorithm, the LSS algorithm rounds K to 0 if cwnd < 25, so it is reasonable to use it as threshold in the second algorithm that is also used as increment in the first algorithm.
With algorithm 1 will be denoted the classical Slow start.

3. Simulation Scenario

The Subject of our analysis will be two four nodes (n0-n3) bottleneck scenarios presented at Figures 2 and 3.

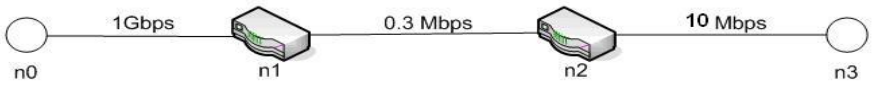

Figure 2. First simulation scenario

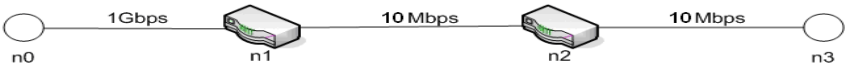

Figure 3. Second simulation scenario

Ns 2.35 is used to simulate the environment. TCP Sack is used as a transport protocol. Additional 20% traffic loses will be included. The possibility of larger initial window will be switched off and on. LSS is used with max_sshtresh set at 100 MSS (maximum segment size). The queuing mechanism is Drop Tail. IFQ receives values of 5, 15, 25, 50, 100 and 200 packets. Link delay is set at 10 ms and the network topology is defined at the Figures 2 and 3. Simulation time is set at 250 s.

4. Simulation Results

The main focus of the analysis will be the throughput and cwnd change. We are going to observe two scenarios both consisted of four nodes. In the first scenario the bottleneck between the nodes n1 and n2 is set at value of 0.3 Mbps and in the second is set at value of 10 Mbps.

4.1 First simulation scenario

At Figures 4, 5, 6, 7, 12 and 13 is presented the cwnd change in accordance of time. Best cwnd behavior at Figure 4 is achieved when algorithm three is used, algorithm two shows slightly better behavior than cwnd achieved when algorithm four and one are used. At Figure 5 cwnd achieved with algorithm 2 and 3 show improved behavior than cwnd achieved with algorithm 1, cwnd achieved with algorithm 4 is slightly shifted. Overlapping of the cwnd

change is achieved when IFQ is set at 25 packets. When IFQ is set at 50 packets best cwnd is provided when algorithm 2 is used. For higher IFQ values like 100 and 200 packets better start up behavior of cwnd is achieved when the improved algorithms are used. The new proposed versions don't differ much in the behavior. If we analyze the throughput behavior shown at the Figures 8, 9, 10, 11, 16 and 17 we can notice improvement in the start up phase. Best throughput when IFQ is set at 5 packets is achieved by the third algorithm, throughput achieved by algorithm's 1 and 4 is overlapping but when the second algorithm is used better throughput is obtained. If we increase the IFQ value at 15 packets than again best start up value is achieved when the second algorithm is used, followed by the throughput achieved with the basic Slow start algorithm, the rest of the algorithms follow. For larger IFQ value of 25 packets the throughput has minor variations during the simulation time. Best throughput is achieved when algorithm two is used followed by the throughput achieved by the forth and the third algorithm and definitely worst throughput behavior in this phase is achieved with the basic slow start. If we increase the IFQ value at 50 packets than the throughput variation is presented at Figure 13. Best throughput in this case is achieved by the second algorithm followed by the basic algorithm. Throughput achieved by the rest of the algorithms is quite similar and is at the last place. If we double the IFQ value at 100 packets than we have similar rank list, first is the throughput achieved with algorithm 2, followed by the algorithm 1, 3 and 4. Doubling the IFQ value at 200 packets will slightly change the rank list where algorithm 2 provides best throughput followed by algorithm 4 than by 1 and algorithm 3. If we summarize the results we can confirm that with the proposed modification of the slow start phase we have achieved improved throughput behavior. If we use Larger initial window algorithm then the results of the achieved cwnd and throughput variations are presented at Figures 14, 15, 20, 21, 22, 23 and 18, 19, 24, 25, 26, 27. At Figure 14 we can see overlapping of the graphs achieved with the algorithms 1, 3, 4. Improved initial behavior is provided by algorithm 2 in the initial phase but lost occurs very fast, mainly caused by the network overflow protection mechanism. cwnd is reduced and equilibrium is achieved. This cwnd behavior affects the throughput as can be seen from Figure 18, worst performance is achieved when algorithm 2 is used and almost overlapping of the throughput charts we have when the other algorithms are used. If we increase the IFQ value at 15 packets the results are shown at Figures 15 and 19. Best cwnd behavior is obtained by algorithm 2 followed by 4, 1 and 3. The throughput charts show similar behavior. With the next IFQ increment at value of 25 packets same cwnd behavior is achieved when algorithms 1 and 3 are used, worst performance for sure is achieved with algorithm 4. We can not define what kind of throughput performances will be achieved when algorithm 2 is used. The throughput presented at Figure 19 shows that best performances are achieved when algorithm 2 is used followed by algorithms 3, 1 and 4. When IFQ is set at 50 packets we have throughput domination when algorithm 2 is used followed by throughput achieved with algorithms 1, 4, 3.

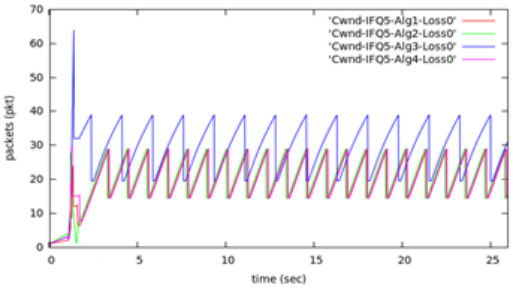

Figure 4. Cwnd Change, IFQ5, Wnd1, Loss 0%

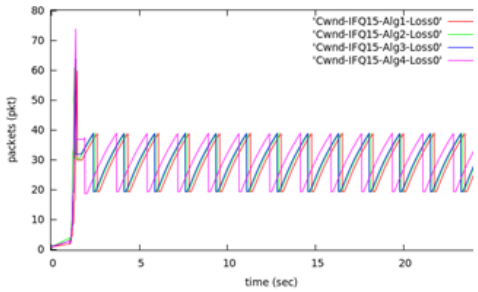

Figure 5. Cwnd Change, IFQ15, Wnd1, Loss 0%

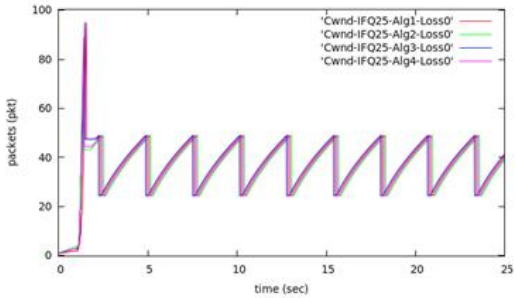

Figure 6. Cwnd Change, IFQ25, Wnd1, Loss 0%

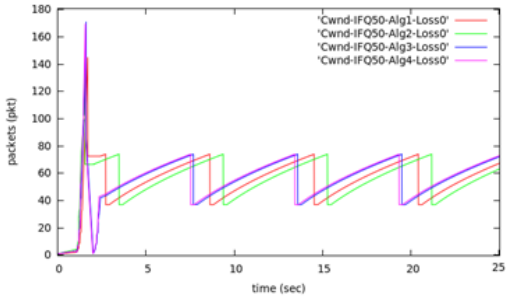

Figure 7. Cwnd Change, IFQ50, Wnd1, Loss 0%

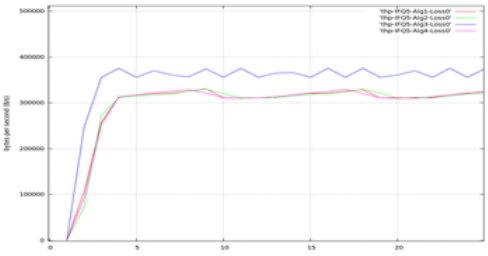

Figure 8. Thp Change, IFQ5, Wnd1, Loss 0%

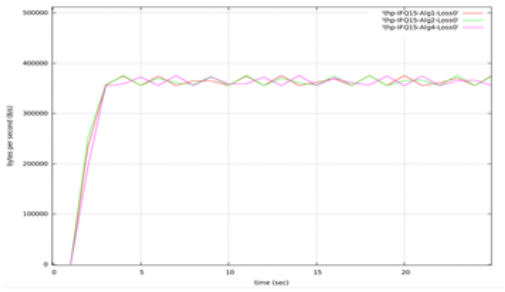

Figure 9. Thp Change, IFQ15, Wnd1, Loss 0%

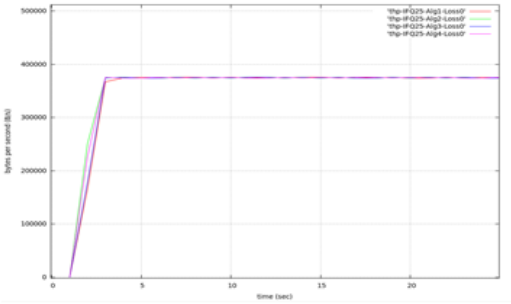

Figure 10. Thp Change, IFQ25, Wnd1, Loss 0%

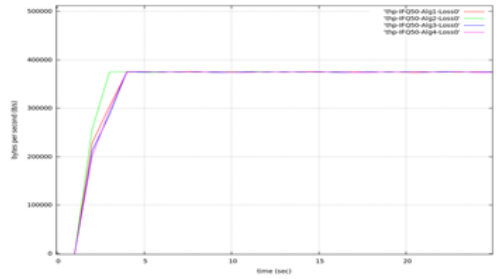

Figure 11. Thp Change, IFQ50, Wnd1, Loss 0%

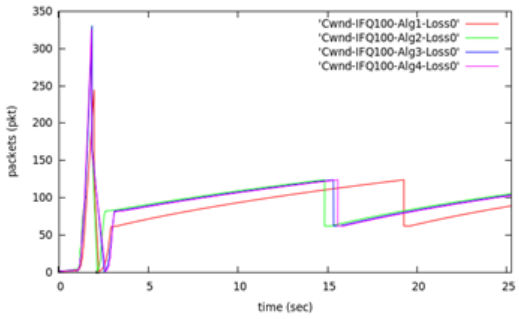

Figure 12. Cwnd Change, IFQ100, Wnd1, Loss 0%

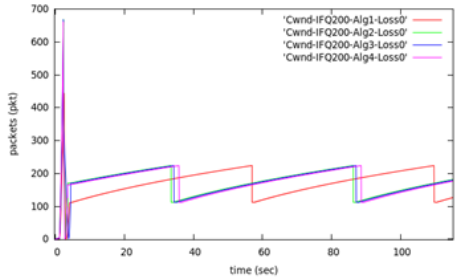

Figure 13. Cwnd Change, IFQ200, Wnd1, Loss 0%

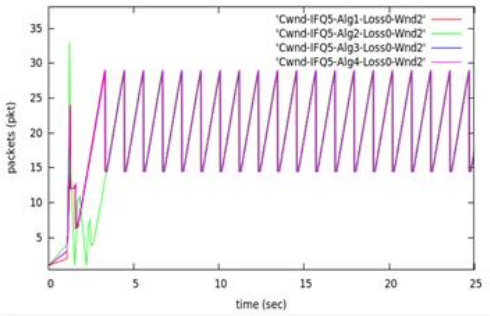

Figure 14. Cwnd Change, IFQ5, Wnd2, Loss 0%

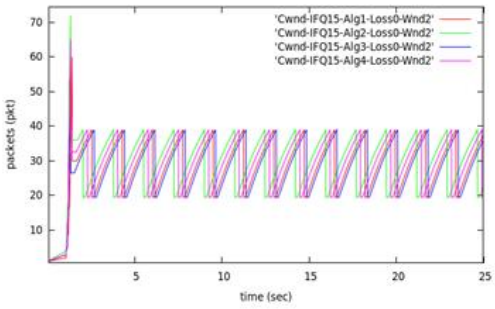

Figure 15. Cwnd Change, IFQ15, Wnd2, Loss 0%

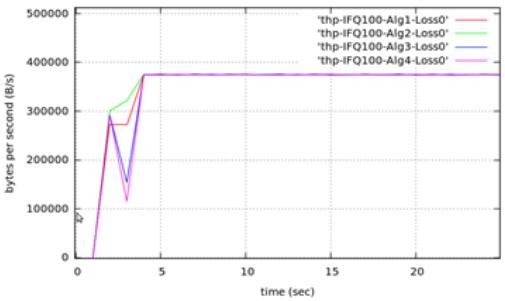

Figure 16. Thp Change, IFQ100, Wnd1, Loss 0%

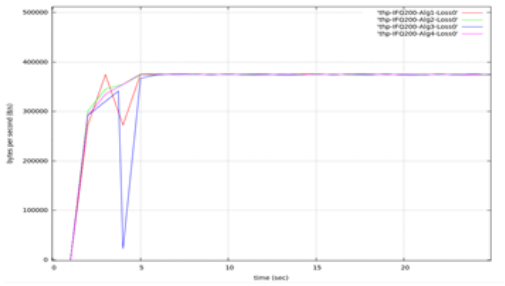

Figure 17. Thp Change, IFQ200, Wnd1, Loss 0%

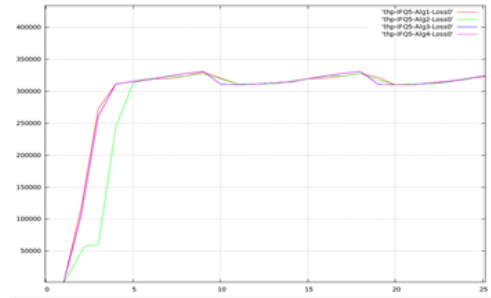

Figure 18. Thp Change, IFQ5, Wnd2, Loss 0%

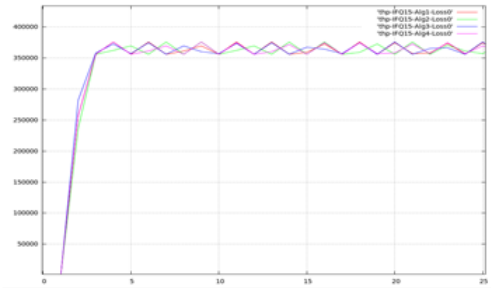

Figure 19. Thp Change, IFQ15, Wnd2, Loss 0%

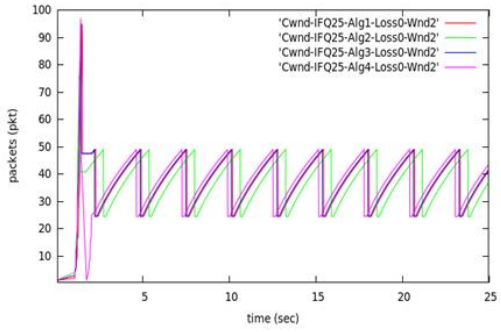

Figure 20. Cwnd Change, IFQ25, Wnd2, Loss 0%

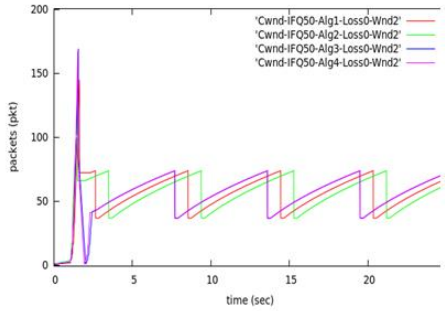

Figure 21. Cwnd Change, IFQ50, Wnd2, Loss 0%

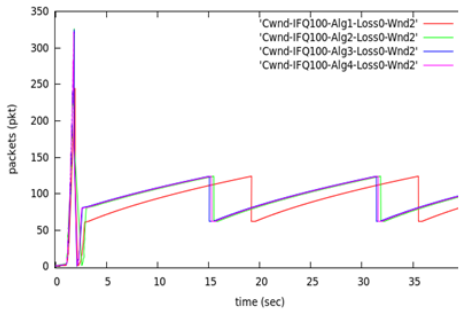

Figure 22. Cwnd Change, IFQ100, Wnd2, Loss 0%

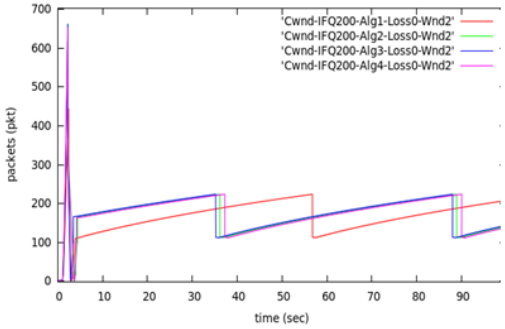

Figure 23. Cwnd Change, IFQ200, Wnd2, Loss 0%

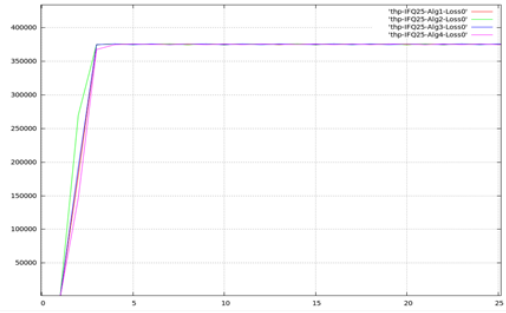

Figure 24. Thp Change, IFQ25, Wnd2, Loss 0%

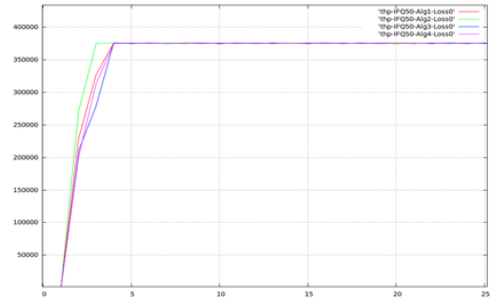

Figure 25. Thp Change, IFQ50, Wnd2, Loss 0%

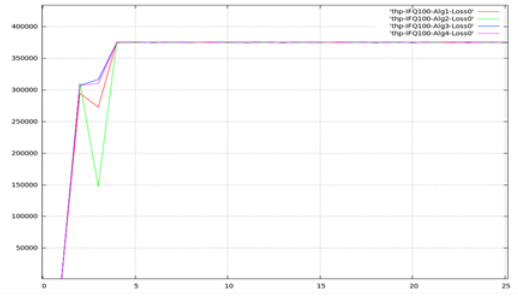

Figure 26. Thp Change, IFQ100, Wnd2, Loss 0%

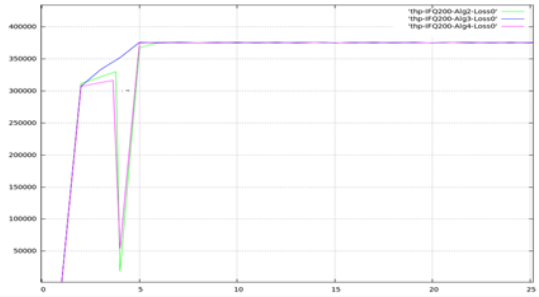

Figure 27. Thp Change, IFQ200, Wnd2, Loss 0%

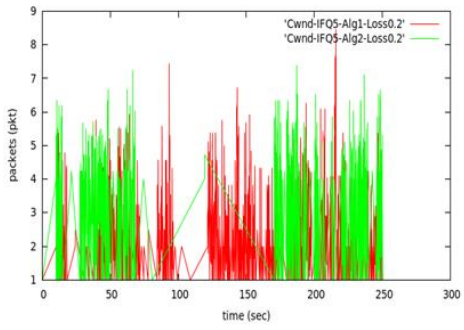

Figure 28. Cwnd Change, IFQ5, Wnd1, Loss 20%

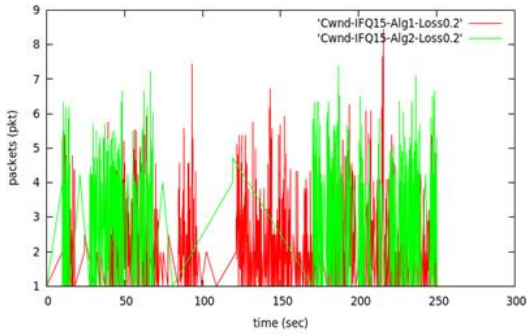

Figure 29. Cwnd Change, IFQ15, Wnd1, Loss 20%

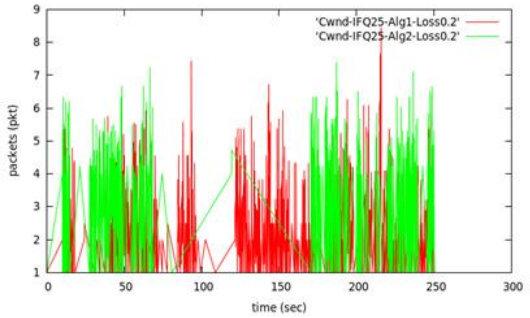

Figure 30. Cwnd Change, IFQ25, Wnd1, Loss 20%

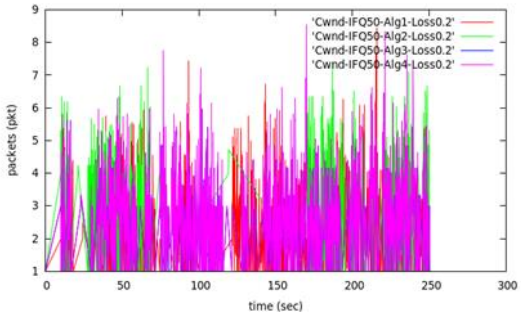

Figure 31. Cwnd Change, IFQ50, Wnd1, Loss 20%

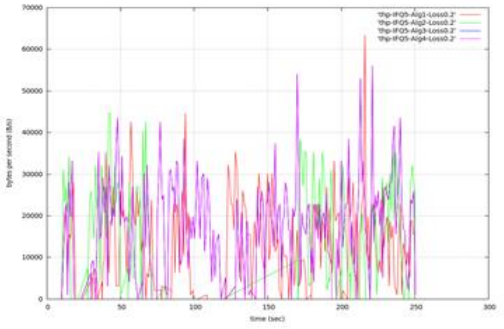

Figure 32. Thp Change, IFQ5, Wnd1, Loss 20%

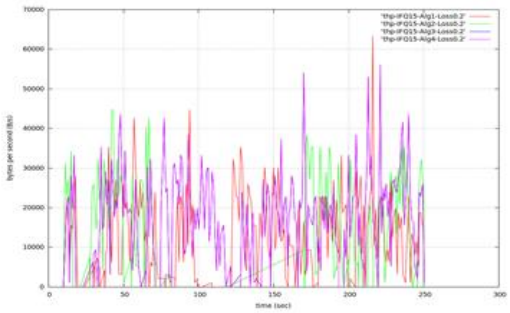

Figure 33. Thp Change, IFQ15, Wnd1, Loss 20%

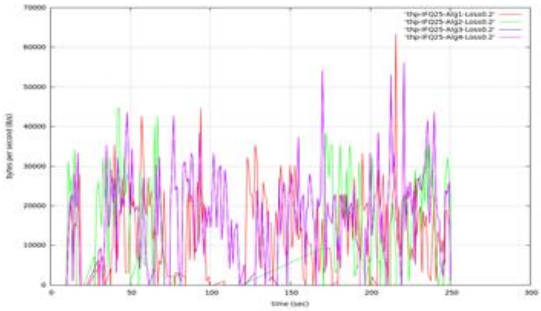

Figure 34. Thp Change, IFQ25, Wnd1, Loss 20%

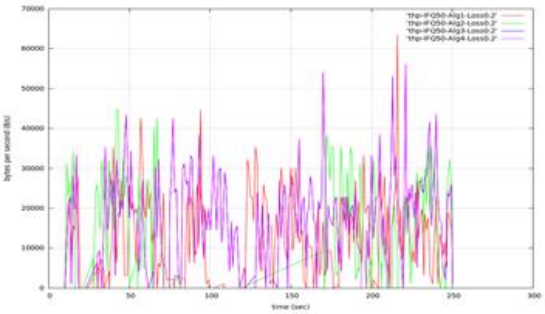

Figure 35. Thp Change, IFQ50, Wnd1, Loss 20%

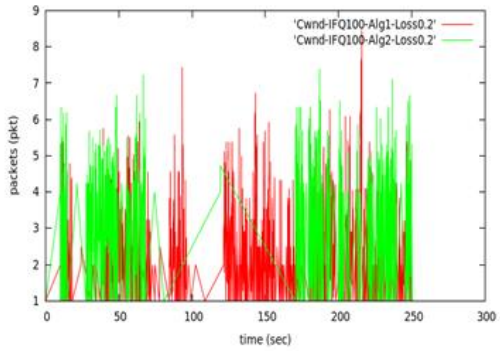

Figure 36. Cwnd Change, IFQ100, Wnd1, Loss20%

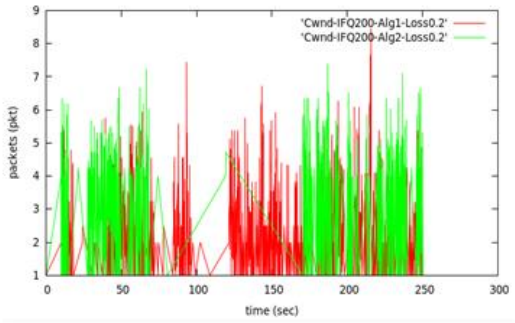

Figure 37. Cwnd Change, IF200, Wnd1, Loss 20%

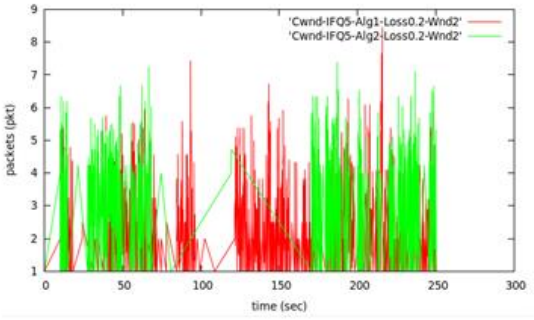

Figure 38. Cwnd Change, IFQ5, Wnd2, Loss 20%

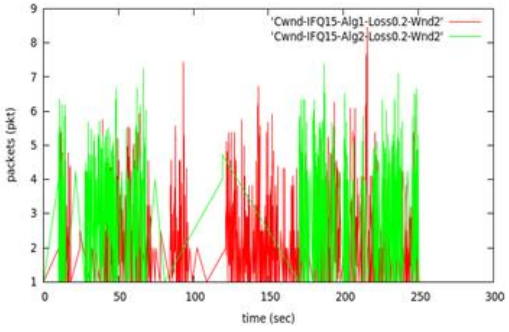

Figure 39. Cwnd Change, IFQ15, Wnd2, Loss 20%

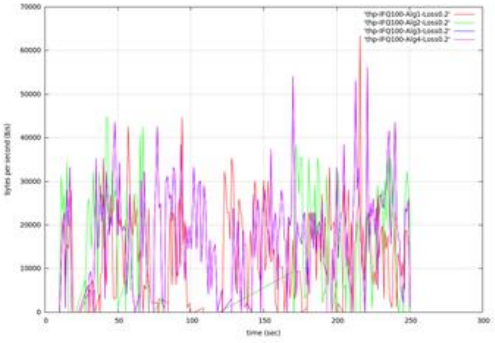

Figure 40. Thp Change, IFQ100, Wnd1, Loss 20%

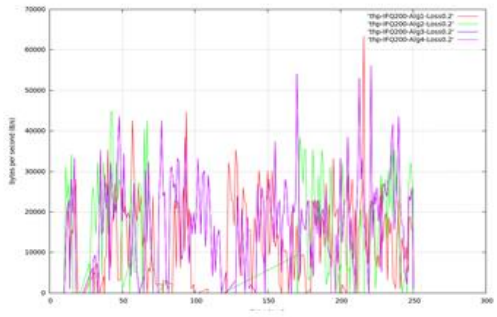

Figure 41. Thp Change, IFQ200, Wnd1, Loss 20%

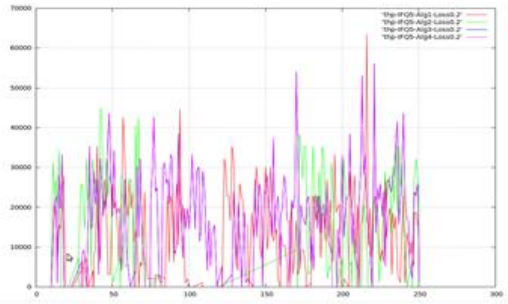

Figure 42. Thp Change, IFQ5, Wnd2, Loss 20%

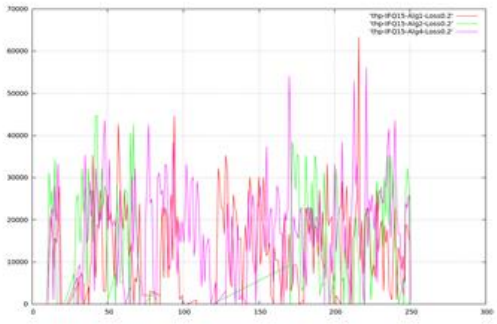

Figure 43. Thp Change, IFQ15, Wnd2, Loss 20%

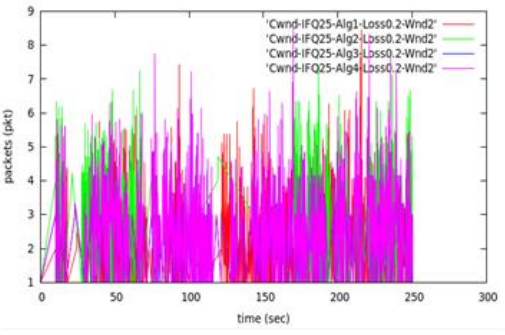

Figure 44. Cwnd Change, IFQ25, Wnd2, Loss 20%

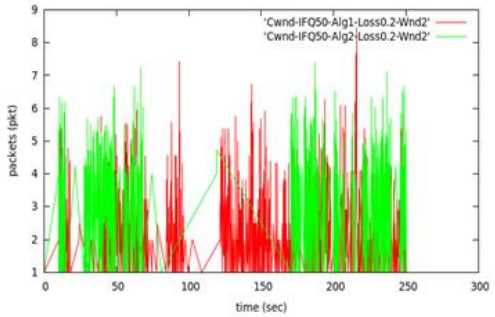

Figure 45. Cwnd Change, IFQ50, Wnd2, Loss 20%

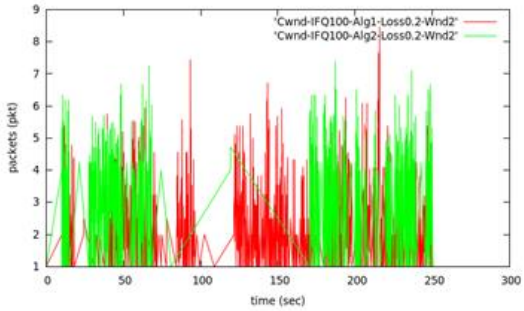

Figure 46. Cwnd Change, IFQ100, Wnd2, Loss 20%

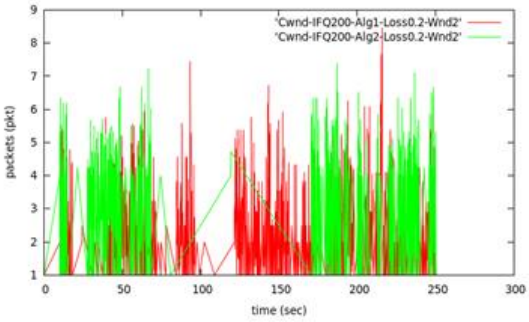

Figure 47. Cwnd Change, IFQ200, Wnd2, Loss 20%

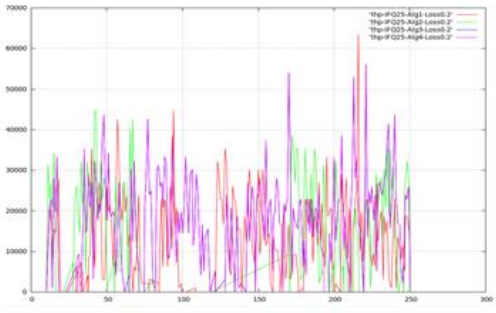

Figure 48. Thp Change, IFQ25, Wnd2, Loss 20%

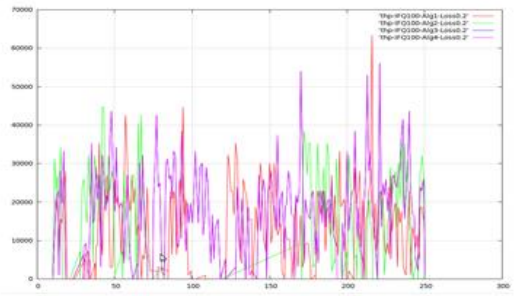

Figure 49. Thp Change, IFQ50, Wnd2, Loss 20%

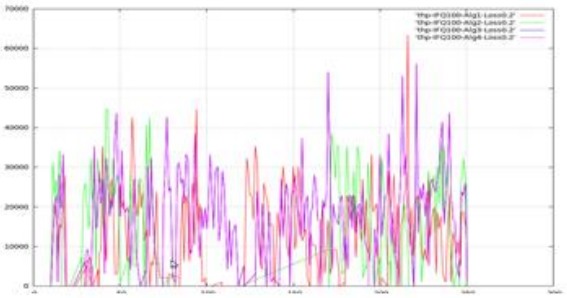

Figure 50. Thp Change, IFQ100, Wnd2, Loss 20%

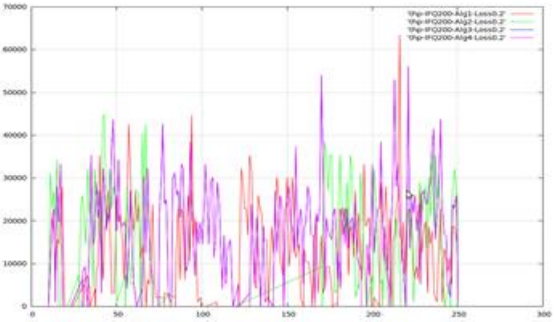

Figure 51. Thp Change, IFQ200, Wnd2, Loss 20%

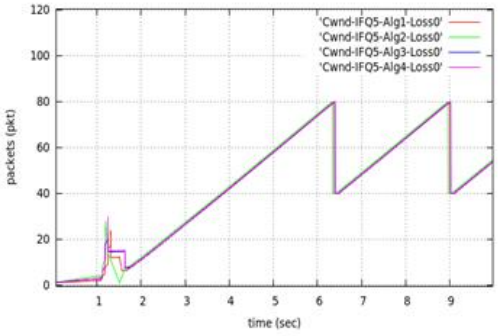

Figure 52. Cwnd Change, IFQ5, Wnd1, Loss 0%

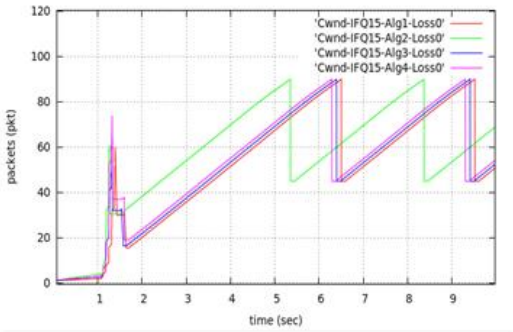

Figure 53. Cwnd Change, IFQ15, Wnd1, Loss 0%

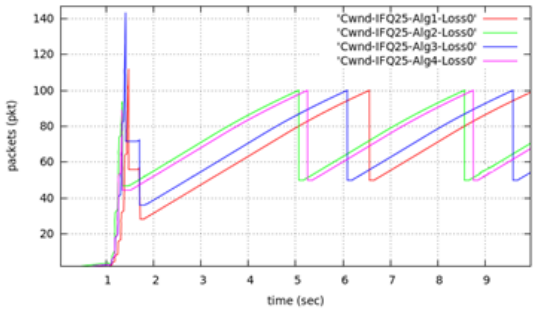

Figure 54. Cwnd Change, IFQ25, Wnd1, Loss 0%

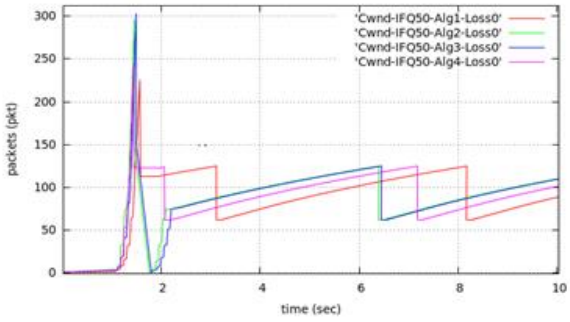

Figure 55. Cwnd Change, IFQ50, Wnd1, Loss 0%

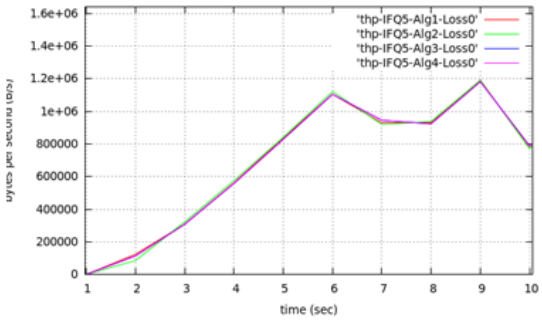

Figure 56. Thp Change, IFQ5, Wnd1, Loss 0%

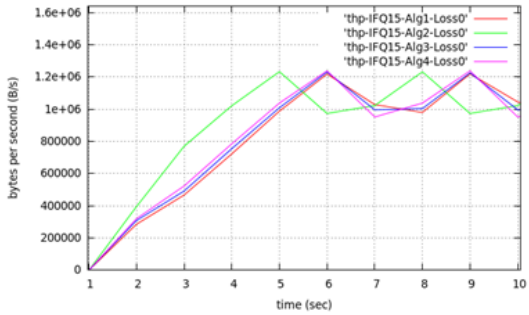

Figure 57. Thp Change, IFQ15, Wnd1, Loss 0%

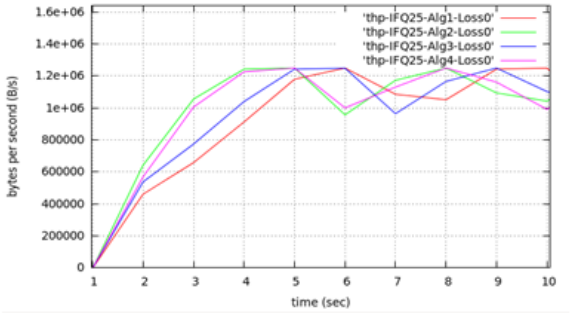

Figure 58. Thp Change, IFQ25, Wnd1, Loss 0%

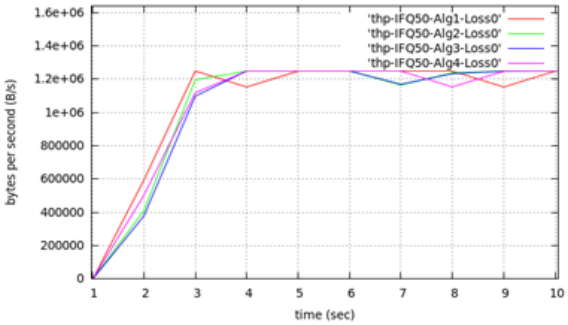

Figure 59. Thp Change, IFQ50, Wnd1, Loss 0%

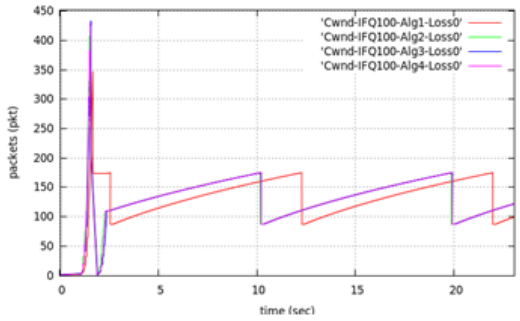

Figure 60. Cwnd Change, IFQ100, Wnd1, Loss 0%

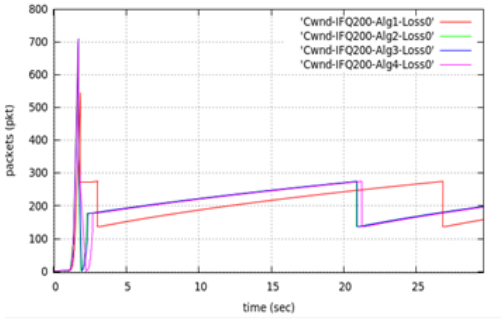

Figure 61. Cwnd Change, IFQ200, Wnd1, Loss 0%

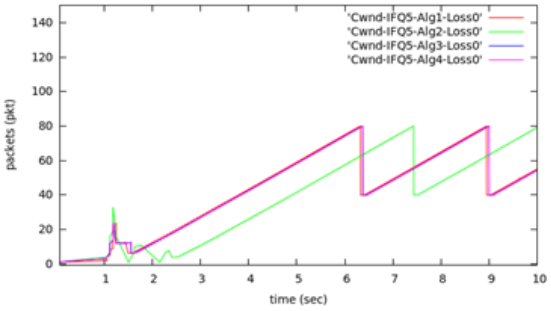

Figure 62. Cwnd Change, IFQ5, Wnd2, Loss 0%

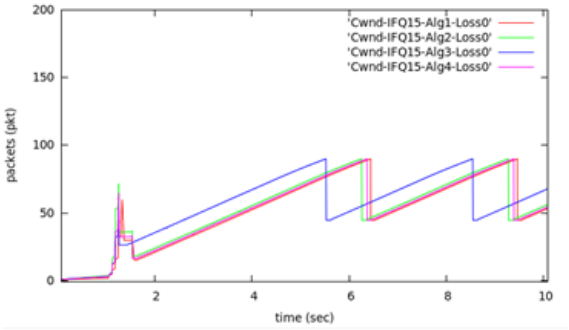

Figure 63. Cwnd Change, IFQ15, Wnd2, Loss 0%

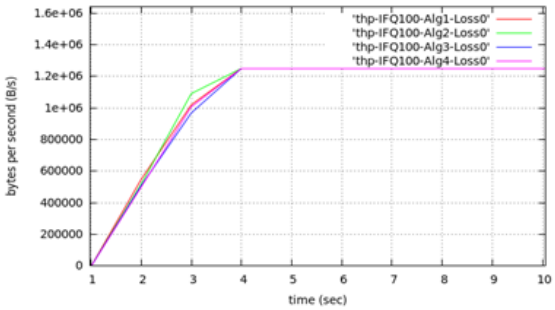

Figure 64. Thp Change, IFQ100, Wnd1, Loss 0%

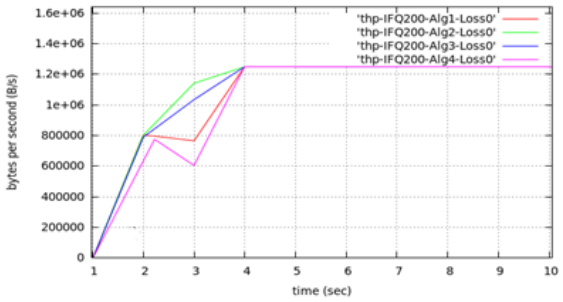

Figure 65. Thp Change, IFQ200, Wnd1, Loss 0%

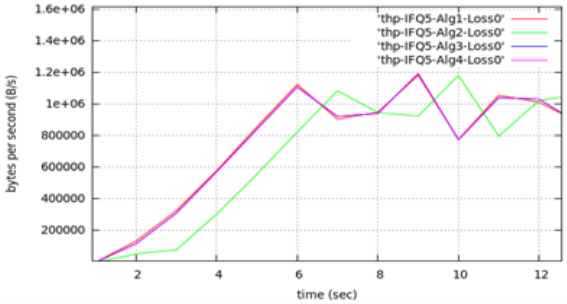

Figure 66. Thp Change, IFQ5, Wnd2, Loss 0%

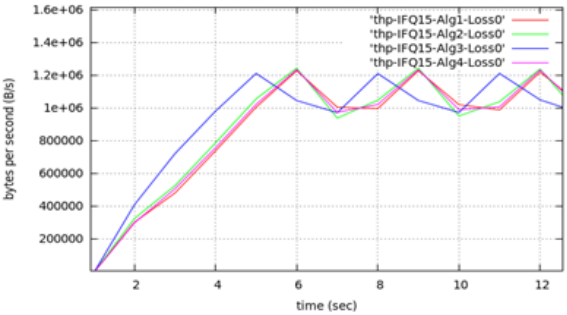

Figure 67. Thp Change, IFQ15, Wnd2, Loss 0%

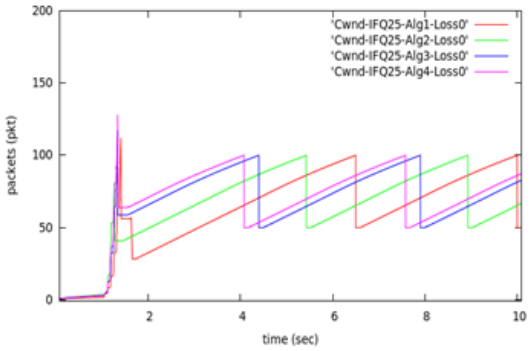

Figure 68. Cwnd Change, IFQ25, Wnd2, Loss 0%

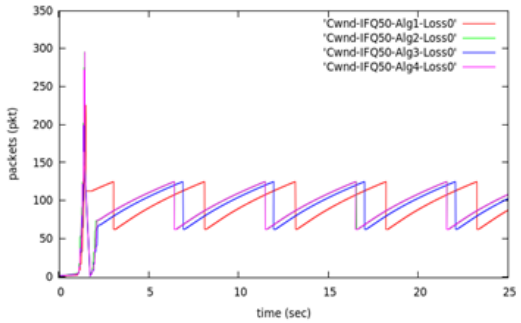

Figure 69. Cwnd Change, IF50, Wnd2, Loss 0%

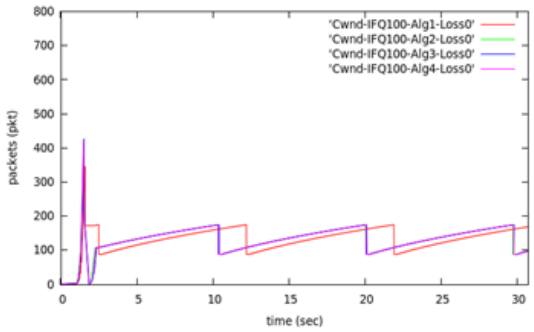

Figure 70. Cwnd Change, IFQ100, Wnd2, Loss 0%

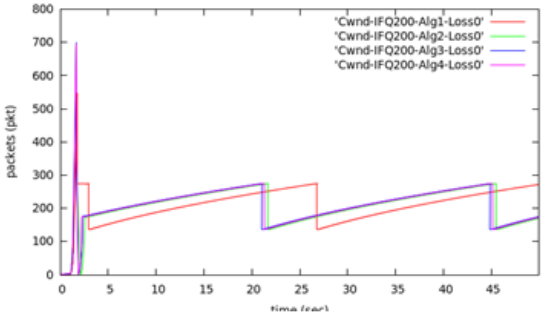

Figure 71. Cwnd Change, IFQ25, Wnd2, Loss 0%

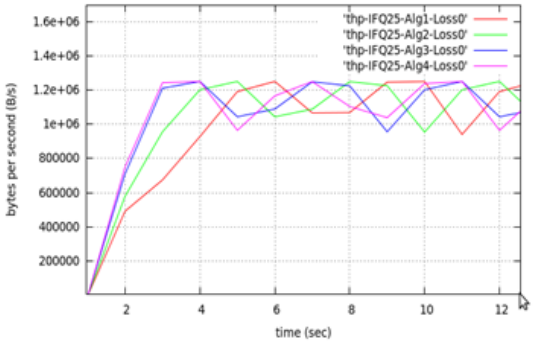

Figure 72. Thp Change, IFQ25, Wnd2, Loss 0%

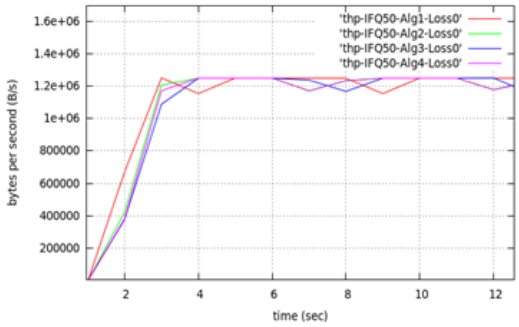

Figure 73. Thp Change, IFQ50, Wnd2, Loss 0%

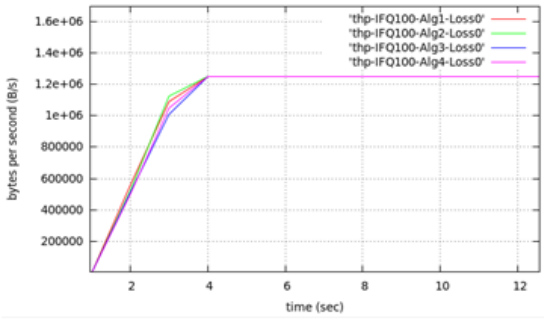

Figure 74. Thp Change, IFQ100, Wnd2, Loss 0%

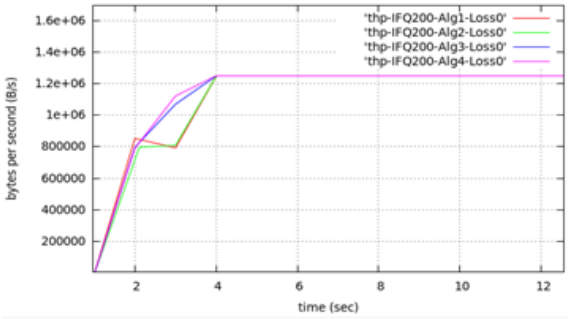

Figure 75. Thp Change, IFQ200, Wnd2, Loss 0%

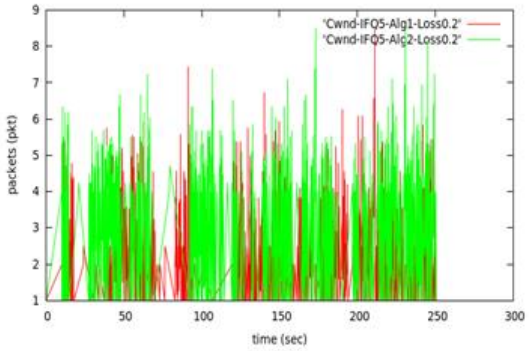

Figure 76. Cwnd Change, IFQ5, Wnd1, Loss 20%

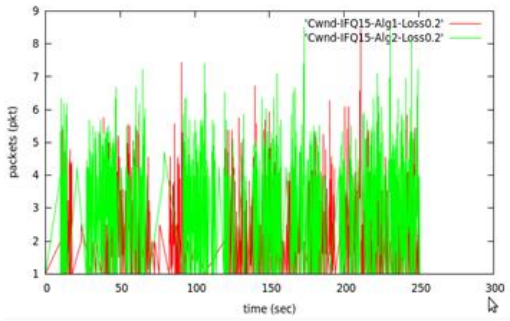

Figure 77. Cwnd Change, IFQ15, Wnd1, Loss 20%

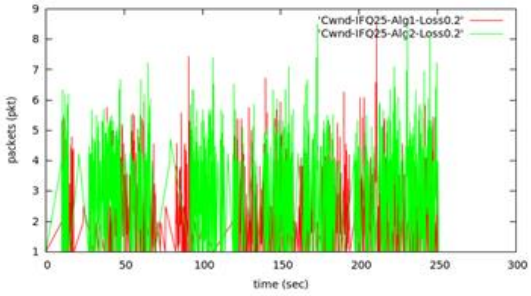

Figure 78. Cwnd Change, IFQ25, Wnd1, Loss 20%

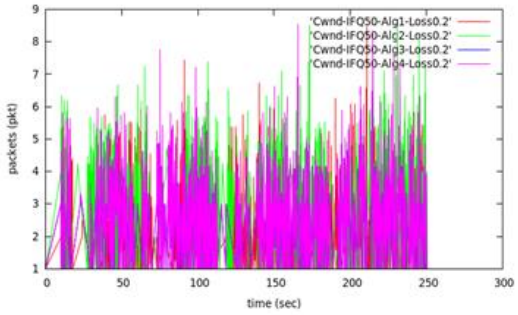

Figure 79. Cwnd Change, IFQ50, Wnd1, Loss 20%

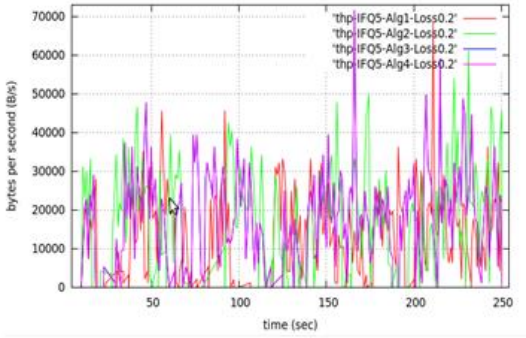

Figure 80. Thp Change, IFQ5, Wnd1, Loss 20%

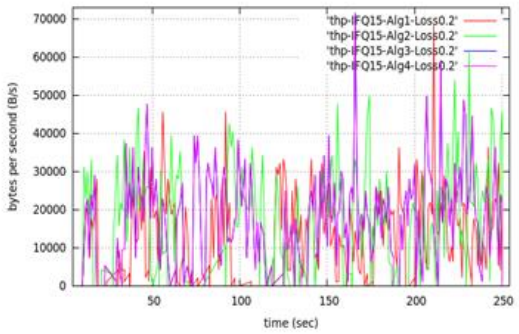

Figure 81. Thp Change, IFQ15, Wnd1, Loss 20%

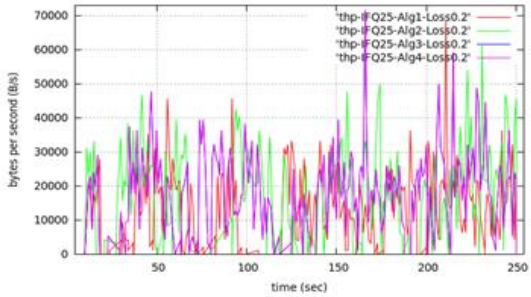

Figure 82. Thp Change, IFQ25, Wnd1, Loss 20%

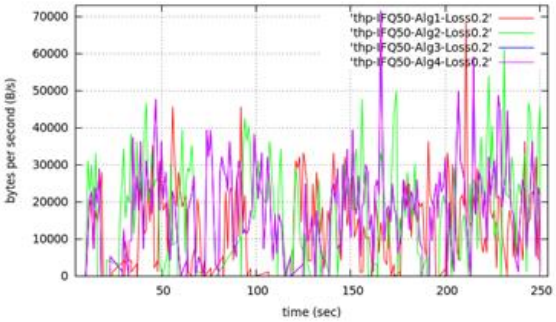

Figure 83. Thp Change, IFQ50, Wnd1, Loss 20%

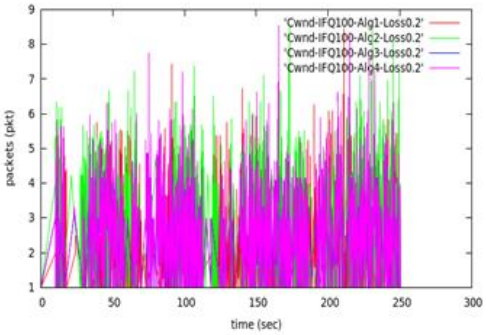

Figure 84. Cwnd Change, IFQ100, Wnd1, Loss20%

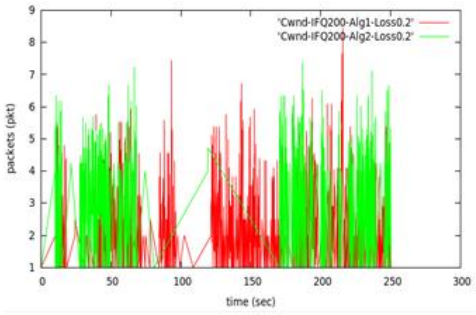

Figure 85. Cwnd Change, IF200, Wnd1, Loss 20%

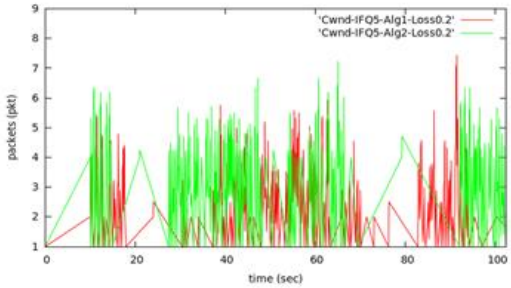

Figure 86. Cwnd Change, IFQ5, Wnd2, Loss 20%

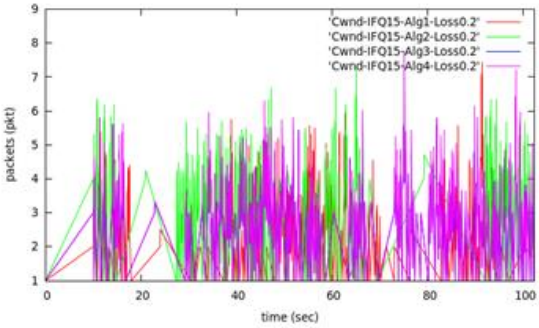

Figure 87. Cwnd Change, IFQ15, Wnd2, Loss 20%

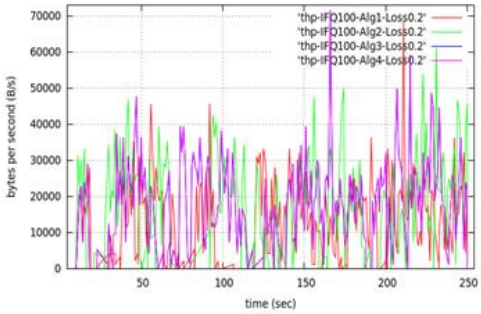

Figure 88. Thp Change, IFQ100, Wnd1, Loss 20%

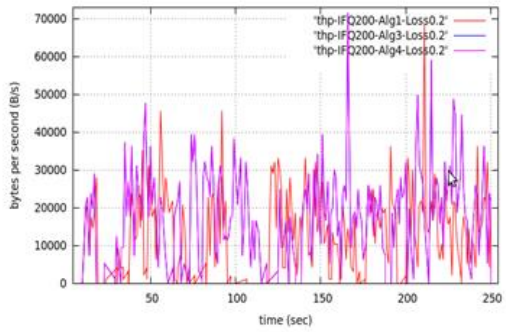

Figure 89. Thp Change, IFQ200, Wnd1, Loss 20%

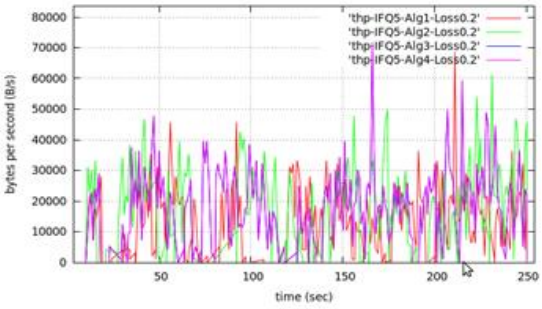

Figure 90. Thp Change, IFQ5, Wnd2, Loss 20%

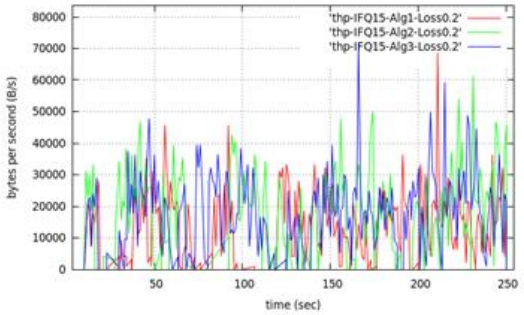

Figure 91. Thp Change, IFQ15, Wnd2, Loss 20%

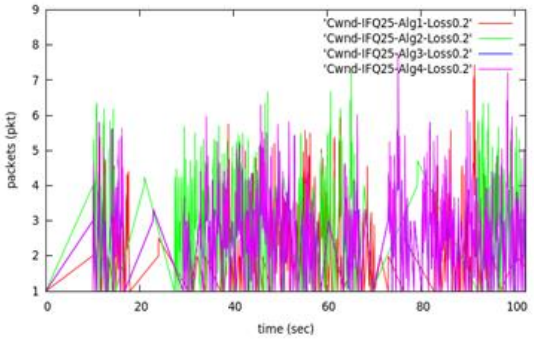

Figure 92. Cwnd Change, IFQ25, Wnd2, Loss 20%

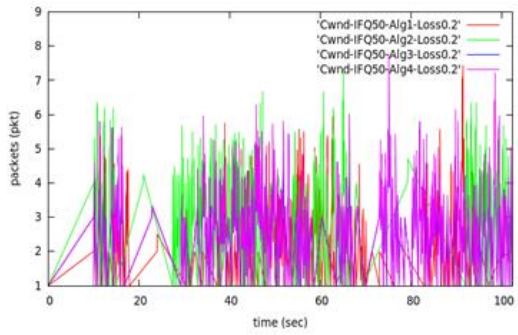

Figure 93. Cwnd Change, IFQ50, Wnd2, Loss 20%

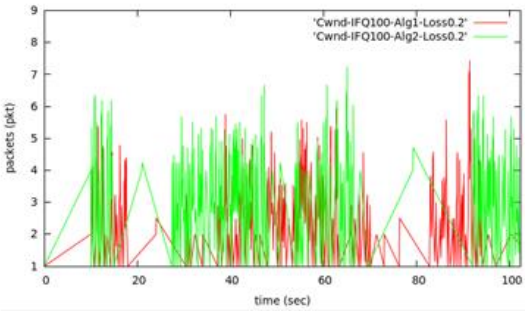

Figure 94. Cwnd Change, IFQ100, Wnd2, Loss 20%

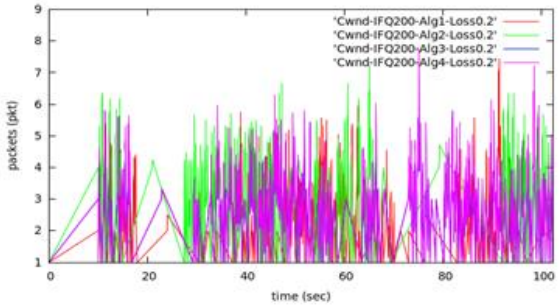

Figure 95. Cwnd Change, IFQ200, Wnd2, Loss 20%

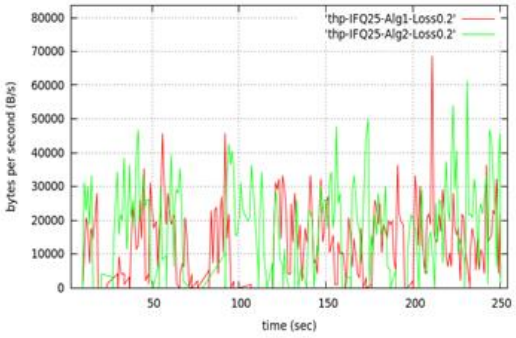

Figure 96. Thp Change, IFQ25, Wnd2, Loss 20%

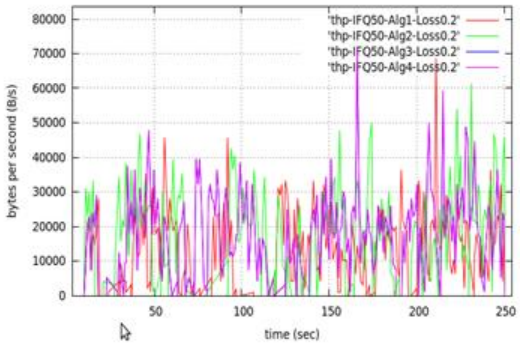

Figure 97. Thp Change, IFQ50, Wnd2, Loss 20%

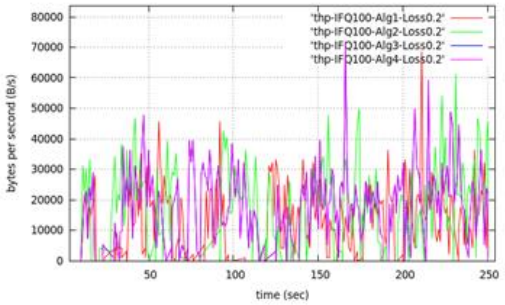

Figure 98. Thp Change, IFQ100, Wnd2, Loss20%

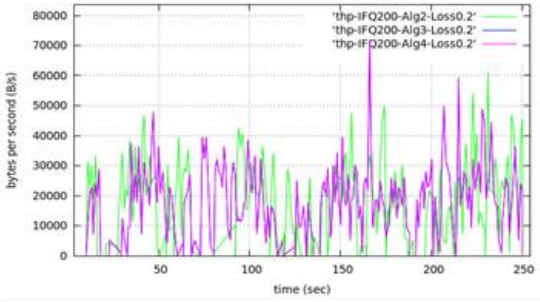

Figure 99. Thp Change, IFQ200, Wnd2, Loss 20%

When IFQ is set at 100 and 200 packets same cwnd behavior is achieved with algorithms 2, 3 and 4 but throughput variation is different, the best is achieved with algorithm 3. From the following we can conclude that for small IFQ values there is no improvement of the throughput. When IFQ is set at values of 25 and 50 packets best throughput is provided by the second algorithm. In case when IFQ receives values of 100 and 200 packets, best performances are achieved when the algorithm 3 is used. If we incorporate 20% throughput losses at the link between n1 and n2 than the cwnd and throughput behavior in case when larger initial window is included/excluded is shown at the figures starting from 28 and ending with Figure 51. We can notice that in both cases with included/excluded larger initial window there are no changes of the cwnd and throughput behavior. We can notice that the size of the IFQ buffer doesn't impact the cwnd and the throughput. One of the possible reasons for this cwnd, throughput behavior is the bottleneck capacity of the link between n1 and n2 and the

incorporated 20% uniform losses. In the initial simulation period cwnd and throughput performances achieved whit the second algorithm outperform the other achieved with the rest of the algorithms this stands for the ending period too. In the middle of the simulation cwnd of the algorithm 2 has large saw shape but it doesn't provide best throughput performances. In the middle of the simulation best throughput performance is achieved with algorithm 3 and 4. It is obviously that they provide identical cwnd and throughput behavior. In this case we can say that they outperform the second algorithm

4.2 Second simulation scenario

The obtained cwnd results of this scenario are presented at the Figures 52, 53, 54, 55, 60, 61. From the figures can be concluded that when IFQ receives values of 5, 15, 100, 200 packets there is almost overlapping of the charts. There is a difference in the behavior only in the initial period of the Slow Start phase. When IFQ is set at 5 packets all algorithms has provided same cwnd and throughput change. If we increase the IFQ size at 15 packets than best cwnd is provided when algorithm 2 is used it enables to be achieved best throughput too. Minor cwnd and throughput variation can be noted when are used the rest of the algorithms. This is not the case when IFQ is set at 25 packets, in this case best cwnd change is obtained with algorithm 2 followed by algorithm 4, 3 and 1, this is the main reason for the throughput behavior presented at figure 58 where grate improvement can be noticed in the slow start phase. If we double the IFQ size than in the initial moment we achieve faster growth of the cwnd function by the advanced algorithms but the bandwidth overshooting limitation requests larger decrease parameter to be used which is the main reason for the rapid degradation of the cwnd value when loss has occurred in the first moments of the slow start phase. This cwnd behavior impacts the throughput and that is why algorithm 1 show best performances followed by the algorithm 2, 4 and 3. If we double the IFQ size again than best throughput will be provided by algorithm 2, almost similar results are obtained when algorithm 1 and 4 are used. Worst throughput behavior is shown when algorithm 3 is used. In the last simulation IFQ parameter receives value of 200 packets. At Figure 61 is presented the cwnd change and at figure 65 is presented the throughput change form where we can notice that best throughput is provided by algorithm 2, second best by algorithm 3 followed by algorithm 1 and 4. If we sublimate the results it is quite obvious that the proposed algorithm provides faster change of the cwnd parameter that enables TCP to speed up the bandwidth estimation process. Larger initial window mechanism provides faster cwnd increase, we will include it in the following analysis. At Figures 62, 63, 68, 69, 70, 71 is presented the cwnd change and at Figures 66, 67, 72, 73, 74, 75 we have presented the throughput change. At the beginning we will set the IFQ value at 5 packets, figures of interest are 62 and 66. Used algorithm 1, 3 and 4 are providing overlapping of the cwnd and throughput graphs. Worst performance in this case we have when algorithm 2 is used. This situation changes if we increase the IFQ at 15 packets. Best cwnd and throughput performances are obtained with algorithm 3 followed by algorithm 2, 4 and 1. When IFQ is set at 25 packets the rank list is as follows algorithm 4, 3, 2, 1. The throughput gain is visible for the improved algorithms. Doubling of the IFQ value changes the situation, algorithm 1 outperforms the rest of the algorithms (2, 3, 4). If we set IFQ at 100 packets, best throughput is obtained when algorithm 2 is used. If we double the IFQ size algorithms 4 and 3 show best performances. From this cwnd and throughput behavior we can conclude that when larger initial window is used and IFQ receives small values the novel algorithms outperform the initial slow start algorithm and provide better throughput utilization except when IFQ is set at 50 packets. Overall we can say that in most of the cases best performances are obtained with algorithm 3 followed by algorithm 2. Usage of the larger initial window mechanism decreases the performances obtained with the algorithm 2 than in

case when this mechanism is not in use. In both scenarios new developed algorithms show better performances in the slow start phase than the initial. We will complete the analysis with incorporating 20% throughput losses between nodes n1 and n2. At Figures 76, 77, 78, 79, 84, 85 is presented the cwnd behavior, the throughput is presented at 80, 81, 82, 83, 88, 89 in case when Larger initial window is not in usage. In this case we have similar situation like in the previous simulation scenario there is almost similar behavior of the cwnd and the throughput regardless the IFQ change. In this case we can say that the IFQ change does not impact the throughput. Same results are obtained when IFQ receives value of 5 packets and 200 packets. During the simulation time of 250 seconds there is domination of the cwnd achieved with the second algorithm. This impacts the throughput and provides best throughput to be achieved when algorithm 2 is used, second best is achieved by algorithm 3 and 4 which behave similar and overlap. If we activate the Larger initial window algorithm then the cwnd and throughput behavior are presented at the following Figures 86, 87, 92, 93, 94, 95 and 90, 91, 96, 97, 98, 99 respectively. Activation of the Larger initial window mechanism resulted with not so vivid cwnd change that is confirmed by the size of the window. In the first 40 s of the simulation we can notice that cwnd is wider, this is repeated when the simulation reaches time of 70 till 90 seconds. In this case best cwnd performances are achieved when algorithm 2 is used followed by algorithm 3 and 4. Throughput follows the cwnd behavior so best throughput is obtained with algorithm 2 followed by the throughput achieved with algorithm 3 and 4. From the figures it is obvious that the throughput obtained with algorithm 3 and 4 is overlapping. Worst performances in this case are shown by the well known slow start, algorithm 1. In this scenario we can say that better performances are achieved with the proposed algorithms than when the link is set at 300Kbps with included 20 % throughput losses and incorporated Larger initial window algorithm.

5. Conclusion

In this paper we have proposed three different algorithms and we have presented the throughput and cwnd behavior. Algorithm 2 is recognized as advanced Slow start algorithm because of his nature, it is fast in the beginning, cwnd increases with value of cwnd=+3 and the decreasing factor is set at 4 after cwnd receives value of 25 the increasing factor is reduced at cwnd=+2 and the decreasing factor is set at 3. In order to provide fast but not too aggressive modification after cwnd receives value of 50 packets we are decreasing the increasing factor of cwnd and it receives values according cwnd=+1 with reducing factor 2 (standard slow start algorithm) after exceeding value of 75 packets, cwnd increasing factor is set at cwnd=+0.5 with decreasing factor set at value of 2, for cwnd values grater than 100 we are using the LSS algorithm. This modification enables TCP to be faster in the initial period but not too aggressive. Variation of this modification is proposed in which cwnd receives value of cwnd=+2 with decreasing factor 3, two variations are tested with threshold parameter set at 25 and 50 packets in order to analyze the aggressive behavior of the proposal. We can say that best throughput behavior in the scenario is provided when algorithm 2 is used so this confirms our decision to name it as advanced slow start algorithm because of it improved characteristics in the initial phase. For smaller IFQ values in both situations we had almost overlapping of the throughput charts especially when the network bandwidth is higher. Increasing of the IFQ size shows more superior behavior of the advanced slow start algorithm, its improved behavior is most recognized for higher IFQ values (100, 200 packets). The other two proposals have more aggressive nature and provide lower performances than the basic slow start algorithm especially for higher IFQ values. If we incorporate Larger initial window in the first simulation scenario than

we can conclude that for small IFQ values like 5, 15 packets all algorithms provide similar behavior. When IFQ receive value of 25 and 50 packets than algorithm 2 outperform the rest of the algorithms but this situation changes when IFQ is set at values of 100 and 200 packets. In this case algorithms 3 and 4 outperform algorithm 1 and 2. Incorporation of Larger initial window mechanism in the second simulation scenario provides overlapping of the throughput chart when IFQ is set at 5 packets, exception is the throughput obtained when algorithm 2 is used, it shows poorest performances. When IFQ receives 15 and 25 packets all new proposed algorithms show better performances than the standard slow start algorithm. Doubling the IFQ value at 50 packets changes the situation and algorithm one shows best performances. We can say that the reason for this behavior is the usage of the Larger Initial Window algorithm and the higher value of the reduction factor in the initial period of the slow start phase. When IFQ is set at 100 and 200 packets again we achieve better performances with the new proposed versions. If we incorporate losses in the system then Larger Initial window and the IFQ change does not impact the throughput and cwnd change in the both scenarios. In the first simulation scenario algorithms 3 and 4 overlap, algorithm 2 show improved performances in part of the simulation time. In the second scenario best performances are obtained when algorithm 2 is used followed by algorithm 3 and 4 which overlap themselves and poorest throughput performances are achieved by algorithm 1. The conducted analysis aloud us to conclude that in lossless systems there is large throughput improvement achieved by algorithm 2, small degradation of the performances is noted when Larger initial window is incorporated this stands when 20% throughput losses are included in case of increased bandwidth between n1 and n2. When are included losses in the first simulation scenario the throughput behavior provided by algorithm 3 and 4 outperform the one achieved when algorithm 2 is used. Generally we can say that the modification is important and algorithm 2 show improved performances in slow and high speed links. IFQ and large initial window mechanism impact the throughput and cwnd change in case of lossless scenario.

References

[1] V. Jacobson, “Congestion Avoidance and Control”, SIGCOMM Symposium on Communications Architectures and Protocols, **(1988)**, pp. 314–329.
[2] V. Jacobson, “Modified TCP Congestion Avoidance Algorithm”, Technical report, **(1990)** April 30.
[3] S. Floyd, RfC 3742: Limited Slow-Start for TCP with Large Congestion Windows, **(2004)**.
[4] C. Caini and R. Ferrincelli, “TCP hybla: A TCP enhancement for heterogeneous networks”, International Journal of Satellite Communications and Networking, vol. 22, **(2004)**, pp. 547–566.
[5] I. Rhee and L. Xu, “Cubic: A new TCP-friendly high-speed TCP variant”, in 3rd International Workshop on Protocols for Fast Long-Distance Networks, Lyon, France, **(2005)** February.
[6] L. Xu, K. Harfous and I. Rhee, “Binary increase congestion control for fast, long distance networks”, in IEEE INFOCOM, 2004, Hong Kong, China, **(2004)** March.
[7] R. Shorten and D. Leith, “H-TCP: TCP for high-speed and long-distance networks TCP”, in Second International Workshop on Protocols for Fast Long-Distance Networks, Argonne, IL, USA, **(2004)** February.
[8] R. Srikant, “The Mathematics of Internet Congestion Control”, A Birkhauser book, **(2004)**.
[9] D. P. Bertsekas, “Nonlinear Programming”, Athena Scientific, 1999. J. Aikat, J. Kaur, F. D. Smith, and K. Jeffay, “Variability in TCP round-trip times,” in 3rd ACM SIGCOMM Conference on Internet Measurement, New York, NY, USA, **(2003)**.
[10] L. A. Grieco and S. Mascolo, “Performance evaluation and comparison of westwood+, new reno, and vegas TCP congestion control”, ACM Computer Communication Review, vol. 34, no. 2, **(2004)** April, pp. 25–38.
[11] BIC and CUBIC protocol - default TCP algorithm in linux, http://netsrv.csc.ncsu.edu/twiki/bin/view/Main/BIC
[12] S. H. Low, L. L. Peterson and L. Wang, “Understanding TCP vegas: A duality model”, in SIGMETRICS/Performance, Cambridge, MA, USA, **(2001)** June.

[13] L. Andrew, C. Marcondes, S. Floyd, L. Dunn, R. Guillier, W. Gang, L. Eggert, S. Ha and I. Rhee, "Towards a common TCP evaluation suite", in 6th International Workshop on Protocols for FAST Long-Distance Networks (PFLDnet 2008), Manchester, UK, **(2008)**.
[14] S. Mascolo and F. Vacirca, "The effect of reverse traffic on TCP congestion control algorithms", in 4th International Workshop on Protocols for Fast Long-distance Networks, Nara, Japan, **(2006)** February.
[15] J. -M. Chen, *et al.*, "Improving SCTP Performance by Jitter-Based Congestion Control overWired-Wireless Networks", EURASIP Journal on Wireless Communications and Networking, **(2011)**.
[16] J. Domzał, N. Ansari and A. Jajszczyk, "Congestion Control in Wireless Flow-Aware Networks", IEEE ICC 2011, Kyoto, **(2011)** June.
[17] H. Jung, *et al.*, "Adaptive Delay-based Congestion Control for High Bandwidth-Delay Product Networks", IEEE INFOCOM 2011, Shanghai, China, **(2011)** April 10-15.
[18] J. Wang, J. Wen, J. Zhang and Y. Han, "TCP-FIT: An Improved TCP Congestion Control Algorithm and its Performance", IEEE INFOCOM 2011, Shanghai, China, **(2011)** April 10-15.
[19] T. Janevski and I. Petrov, "Impact of Duplicated ACK on TCP Performances in Wireless Chain Environment", International Journal of Advanced Science and Technology, vol. 47, Korea, **(2012)** September.

Authors

Ivan Petrov is with telecom operator Makedonski Telekom, Skopje, Macedonia. He received his Dipl.Ing. and M.Sc. from the Faculty of Electrical Engineering, Ss. Cyril and Methodius University in Skopje, in 2005 and 2009, respectively. From 2009 he is working towards his Ph.D. degree under mentorship of Prof. Toni Janevski. His research interests include transport protocols and cross-layer optimization techniques in heterogeneous wireless IP networks.

Dr. Toni Janevski is a Full Professor at the Faculty of Electrical Engineering and Information Technologies, Ss. Cyril and Methodius University in Skopje, Macedonia. He received his Dipl. Ing., M.Sc. and Ph.D. degrees in electrical engineering all from Faculty of Electrical Engineering and Information Technologies,

Ss. Cyril and Methodius University in Skopje, in 1996, 1999 and 2001, respectively. In the past, during 1996-1999 he has worked at the Macedonian mobile operator Mobimak (currently T-Mobile, Macedonia), contributing to the planning, dimensioning and implementation of the first mobile network in Macedonia. From 1999 he is with Faculty of Electrical Engineering and Information Technologies in Skopje. In 2001 he has conducted research in optical communication at IBM T. J. Watson Research Center, New York. Also, during 2005-2008 he was a Member of the Commission of the Agency for Electronic Communications (AEC) of the Republic of Macedonia, where he contributed towards the introduction of new technologies in Macedonia, such as WiMAX and 3G mobile networks, as well as new operators and services. During the periods 2008-2012 and 2012-2016 he is an elected member of the Senate of the Ss. Cyril and Methodius University in Skopje. In 2009 he has established Macedonian ITU (International Telecommunication Union) Centre of Excellence (CoE) as part of the Europe's CoE network, and serves as its Coordinator since than. He is an author of the book titled "Traffic Analysis and Design of

Wireless IP Networks", which is published in 2003 by Artech House Inc, Boston, USA. Also, he is author of the book "Switching and Routing", written in Macedonian language, published in September 2011 by the Ss. Cyril and Methodius University in Skopje, for which in May 2012 he has won the "Goce Delchev" award, the highest award for science in the Republic of Macedonia. He has published more than 130 research papers and has led research and applicative projects in the area of Internet technologies and mobile and wireless networks. He is a Senior Member of IEEE. His research interests include Internet Technologies, Mobile, Wireless and Multimedia Networks, Traffic Theory, Quality of Service, Design and Modeling of Telecommunication Networks, as well as Next Generation Networks.