APTIKOM Journal on Computer Science and Information Technologies
Vol. 3, No. 1, 2018, pp. 13~20
ISSN: 2528-2417
DOI: 10.11591/APTIKOM.J.CSIT.81

Development of an efficient mechanism for rapid protocols using NS-2 simulator

Sarah N. Abdulwahid
Scholarships and Cultural Relationship Directorate Ministry of Higher Education and Scientific Research, Baghdad, Iraq

Abstract

The delivered effort in this manuscript is grounded on NS-2 (The Network Simulator 2) to implement the congestion control process of classic TCP (Transmission Control Protocol), with new congestion control mechanism. In this paper, a novel congestion control algorithm is offered, which contains of slow-start and congestion avoidance mechanisms. The proposed slow-start algorithm assumes a duplicating and an interpolating approach to the congestion window (cwnd) for each increment instead of the exponential increment used by other TCP source variants such as Reno, Vega, Tahoe, Newreno, Fack, and Sack. Furthermore, the enhanced congestion avoidance algorithm is built by using an improved Additive Increase Multiplicative Decrease (AIMD) algorithm with multi TCP flow facility, to provide an enhanced congestion control algorithm with some valuable properties to improve TCP routine for high speed protocols. The improvement strategy based on merging of slow start, congestion avoidance mechanism that are used in TCP congestion control, to create a new AIMD algorithm with a new relationship between the pair parameters a and b. This paper is also involved in the creation of rapid agent in NS-2 models designed to identify the modified TCP and to configure the NS-2 platform. A fast TCP also includes an innovative scheme to slow the rapid start to help TCP to start faster through the high speed networks and also to postpone the congestion state as much as possible.

Keywords: *congestion control, NS-2, TCP*

1. Introduction

Essentially creating a new TCP agent in NS-2 [1] needs to improve new congestion control tool or toincrease the stage of a slow start or a stage of congestion avoidance, by adjusting the parameters to adjust the behaviour of congestion window such as the slow start threshold (ssthresh). The networks include the present generations and future generation, high-speed connections and are mostly with low-propagation delay, as it is required to run over a new set of protocols thatcan provide equitable act on these speedy links. Then, the techniques used in the classic source TCPs cannot be relied upon to perform well across wired and high-speed wireless networks, also, the standard algorithm (exponential such as slow start) should be speedier. This forces the designers to propose new mechanisms with high performance, which will be reliable for the completion of the transition process in the shortest time and with a little packet loss or queuing. Despite ofa significant impact on congestion control is strengthened due to improved phases of slow start or congestion avoidance, but there are a lot of versions of TCP arebased on enhancingcongestion control with a new modification in two parts. This methodology can provide a robust and strict mechanism for adjusting the cwnddue to the slow start and congestion avoidance practically work gathering and intervention in process including control of the congestion window. Therefore, extensive modification and promoting technical cooperation program proposed for these key algorithms that are used to achieve high-speed protocol and covered dominated congestion control. The technical interpolation scheme is used to push the window to grow faster; with approximating cwnd size via quadratic and polynomial interpolation to reach the ssthresh of the network connection. In addition, in congestion avoidance phase, the enhanced TCP includes a novel strategy to enhance the TCP performance over high speed environments. The proposed mechanism also includes a facility to send multiple flows over same connection with a novel technique to dynamically estimate the number of available flows.

TCP-Tahoe supports slow start procedure, which is the first step in themechanism of congestion control [2]. The needing to create a transmission network with TCP to gently investigate the network for regulating network bandwidth can be reached to avoid network congestion with a great burst of packets. This

Received December 11, 2017; Revised February 10, 2018; Accepted February 21, 2018

problem is solved by pushing multiple packets in a slow start at the beginning of the transfer procedure, or when transmission of data is characterized by loss of synchronization relay. Through a slow start, the new TCP increases cwnd received withmostly maximum clip size (MSS) bytes per packet an acknowledgment (ACK), which recognizes them, and the phase completes a slow start, when ssthresh exceeds the level or when acongestion is noticed. So, the contribution and the motivation of this research deals with a development a new TCP with new specifications to fulfil the requirements of high speed networks and the new cellular systems by enhancing the slow-start mechanism to send more packets before the congestion state and also by avoiding the congestion situation via improving the AIMD algorithm with multi TCP flow facility instead of providing one flow. The structure of this paper will be as follow: Section II explainsthe formation of the new slow-start mechanism of TCP, Section III discusses the formation of congestion avoidance technique, Section IV, presentsmerging and combining the improved slow-start and congestion avoidance mechanisms to deliver the final congestion control technique, Section Villustrates the required modification in network simulator 2 to implement the new congestion control algorithm and finally the discussion and conclusion of the obtained algorithm will be in Section VI.

2. Formation of TCP Slow-Start Algorithm

The new algorithm provides primary concept of a slow start algorithm cwnd where TCP sender cannot send packets more than the data size that the cwnd is recognized in the pipe of network. By adjusting cwnd, TCP is tuning transmission rate in the way of communications. The slow start phase includes two objectives: the first, when a new connection is established, does not determine the value of cwnd and uses an algorithm initially slow start to identify [3]. In addition, the TCP protocol is to cut auto-synchronized, where is the use of supported packages and around the clock to select the signal to the next link in the packages [2]. When there is no packet on the pipeline, which is a new situation for the link or after the deadline there is no confirmation message service access flashes [4]. Then it will use a slow start to gradually increase the number of packets to be sent in the beginning. In the suggested slow start, the process is separated into two separated phases. The first stage is used to rapidly rise the cwnd on window, while the other customs the stage to approach ssthresh with steps of a small increasing [5]. In (1) we provide an algorithm of slow start with full principle, in which the complete algorithm as a set of plans to increase in both phases, including the use of a doubling of growth if cwnd is less than ssthresh/2, while the furtherincreasing in the use of Lagrange polynomial interpolation, and when the situation becomes effective [6].

$$cwnd(t+\tau) = \left\{ \begin{array}{ll} 2 * cwnd(t) & \text{if cwnd(t) <ssthresh/2} \\ \dfrac{2 * ssthresh * cwnd(t) - cwnd^2(t)}{ssthresh} & \\ \text{if (ssthresh/2 >= cwnd(t) <ssthresh)} & \end{array} \right\} \quad (1)$$

The suggested process grounded on the innovative algorithm and the exponential algorithm of typical slow start but is divided into two phases as described above. The main stage begins cwnd when the increase in value and initial increase while it less than ssthresh/2. When more refined, but cwnd reaches the ssthresh that begins to control the second phase to reduce by the use of the linear formula. In this technique, which is always increasing cwnd per ACK received, but the value of increasing becomes smaller when it reaches ssthresh of cwnd. The increasing is stopped at the time when these two phases is reached the ssthresh, then the congestion control moves to the stage of congestion avoidance. The routine of the proposed algorithm can be written as in the following steps:

Step 1: Initialization: Set: *cwnd*=1, initial *ssthresh* = *ssthresh* (initial value),
Step 2: Send packets using the window size *cwnd.*
Step 3: for each new received ACK, Go to step 4;
Step 4: if ($cwnd \leq ssthresh/2$), then *cwnd*=2 x *cwnd* then Go to step 2;
Step 5: if (*ssthresh*/2 <*cwnd*&&*cwnd*<*ssthresh*), then *cwnd*=2 x *cwnd*-(*cwnd*)^2 /*ssthresh* then Go to step 2;
Step 6: if retransmission timeout occurs or receive three DUBACKs, then *cwnd*=*ssthresh* and Go to step 2;
Step 7: if (*cwnd*>*ssthresh*), then enter the congestion avoidance phase

3. Formation of TCP Congestion Avoidance

The TCP congestion control algorithm contains a congestion avoidance process too, which comprises ofthree phases, the first phase of public AIMD to adjust the mounting losses and toreduce the congestion windows every RTT [7]. The relationship between merged parameteris enhanced by deriving a and b parameters to improve the association that can offer additional essential paths on the equivalent TCP connection. The new association gives a wide series of requiredvalues of a and b thatallowing to the number of virtual routesto be supposed by the user, which denotes the next stage to improve the algorithm to avoid congestion. Consequently, AIMD (a, b) customs a new association among a and b, grounded on the quantity of virtual flows of TCP, nevertheless must set these flows by the user withthe ability to adjust it dynamically. Thus, the concentration of the third phase of the strategy and employing a new procedure to appraisethe number of available streams through the network pipe, to get the concentrated data transmission from the source to the destination by TCP streams (k). At that time, the amount of flows growsdepending on network situations and when these conditions are allowed to use 2.3, 4, or more of the technical cooperation program, if we use multiple TCP connections on the same link. The final arrangement algorithm to avoid congestion, before the implementation of AIMD, it is essential to optimal number of virtual TCP estimate flows according to the size of the previous and current congestion window size [8].

$$k = \frac{1}{2}\left[\frac{cwnd_t}{cwnd_t - cwnd_{t+\tau}}\right] \tag{2}$$

The value of k, should be estimated to form this relationship afterspotting congestion situation or accurately after every packet losses happening. Then it will formulate estimation according to the algorithm as in the following steps:

```
Initialization, $k$=1 and $f$=0;
Step 1: When packet loss occurs, enter $k$ estimation routine, Go to Step 2.
Step 2: k=0.5 ($f/(f-cwnd)$) ;
Step 3: keep the last value of $cwnd$, $f=cwnd$ ;
Step 4: Stop $k$ estimation routine and Go to congestion avoidance (Subroutine c);
```

where *k* is the number of virtual flows and *f* represents a temporary variable to store the last window size after last packet lost.

$$b = \frac{2}{1+3k} \tag{3}$$

Applying AIMD (a,b) is based on the increase in the cwnd estimate ofthe default flows, the association between the pair parameters a and b, so when = k, increases the cwnd by k/cwnd in RTT whole size of cwnd become:

$$cwnd = cwnd + (k / cwnd) \tag{4}$$

In the event of a loss, the cwndis adjusted by the parameter b, wherever b is considered from the improved relations between a and b, as well as restricted by the amount of virtual k flows, wherever the ending formulation of increasing in the event of packet loss becomes as follows:

$$cwnd = cwnd - \left(\frac{2}{3k+1}\right) * cwnd \tag{5}$$

Conforming to the previous two formulations, needs k to be estimated with every packet loss to adjust the cwnd size and if there is any loss, theincreasing in cwnd will still grow by k/cwnd every RTT. The construction of the congestion avoidance algorithm includes the following procedures:

```
Step 1: For each RTT, $cwnd = cwnd + (k/cwnd)$; Go to step 2;
Step 2: If loss occur, $cwnd = cwnd - (2/(3k+1))\ cwnd$; Go to Step 3;
Step 3: Estimate new value of $k$; Go to step 1;
```

It is significant to be annotated that the approximation of k must be done afterward the loss occurs, to avoid banned values, and there is no risk if k become negative or does not matter if k is not an integer value.

4. Combined Congestion Control Mechanism

The merging of slow start, congestion avoidance algorithm, and estimating the optimum number of flows represents a virtual control congestionsystem, which proposes a novel control onthe congestion for cwnd TCP window. In details, the suggested TCP application and algorithm of congestion control on TCP Reno to getthe benefit from the rapid recovery algorithms and rapid re-transmissions, which are previouslyconvoluted in congestion control of Reno TCP as presented in the graph of state transition diagramin Figure 1.

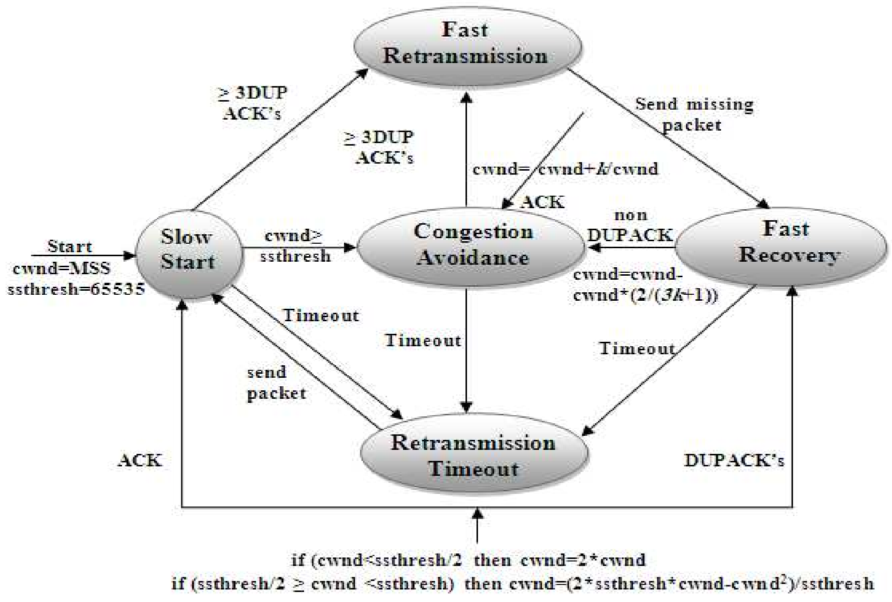


Figure 1. State transition diagram of proposed congestion control mechanism

TCP- Reno requests immediate ACK afterreceiving a segment from the destination. The reason behind that, when the source accepts ACK (DUPACK), then this DUPACK may have established, if successful segment in the slice predictable sequence has been delayed in the network pipeline and theaccessed segments are out of order or in the situation of lossoccurrence in the packets. Also, when the source receives a number of DUPACK, this means that it is the right time to be approved and even if there was a segment has taken the extended path, then itmust be obtained by the destination now. In high speed wireless network, there is a high chance that the packet loss happened, then TCP-Reno algorithm suggests quickretransmission and when the sender accepts three DUPACK, this means that the slide-borne interrupted, then you must resend the missing part without waiting till the deadline. Another reason for choosing TCP- Reno is that after the loss of part flow, Reno does not reduce the cwnd to one part because it will make an empty networkpipeline.

5. Required Modifications in NS-2

The acceptance of afast TCP as a new protocol inNS-2 simulator, requires performing modifications on the source files to generate (or edit) the TCP agent called RTCP and for making new TCP ready for the integration with a new algorithm to control congestion. Inappropriately, the processes of adding RTCP in NS-2areprecise and complicated, becausethere is no expert documents for these actions and other threats. For instance, NS-2 does not contain the complete help in the procedures which are written in assembly (all necessary changes are done using C++), and when developers face fault, they should review all of the steps that have been made. Files that have been modified in question here aregrounded on the version 2.3X, then the suggested TCP described the rapid supposed to be tried more than NS-2.3Xseries.

1. The first modification applied on **ns-compact.tcl** file in the location */ns-allinone-2.3x/ns-2.3x/tcl/lib/* by adding the single line as shown below:

```
$self map_ns_defaultsns_rapidtcp
```

Then, adding the statements below in the correct place in same file:

```
# Agent/TCP/Rapid
    TclObject set varMap_(rampdown) rampdown_
    TclObject set varMap_(ss-div4) ss-div4_
```

In addition, it is necessary to add the couple of statements shown below:

```
set classMap_(tcp-rapid) Agent/TCP/Rapid
    set classMap_(rapidtcp) Agent/TCP/Rapid
```

2. Then, the file **Makefile.in** that located in: */ns-allinone-2.3x/ns-2.3x/Makefile.in* needs to add a single statement in the same groups of other TCP variants as expressed below:

```
tcp/tcp-rapid.o
```

3. Other single line should be added to the file **ns_tcl.cc** located in: */ns-allinone-2.3x/ns-2.3x/gen/ns_tcl.cc* as following:

```
$self map_ns_defaultsns_rapidtcp\n\
```

4. Two short statements should be added to the file **FILES** in the main NS folder *ns-2.3x* as stated below:

```
tcp/tcp-rapid.cc
tcp/tcp-rapid.h
```

Additionally, in the same file, the next four lines must be added as shown below:

```
       tcl/test/test-output-tcpVariants/fourdrops_rapid.gz
       tcl/test/test-output-tcpVariants/onedrop_rapid.gz
       tcl/test/test-output-tcpVariants/threedrops_rapid.gz
       tcl/test/test-output-tcpVariants/twodrops_rapid.gz
```

5. In the location:*/ns-allinone-2.3x/ns-2.3x/tcp* the file **tcp-fs.h** requires to involve the identification routine of TCP Rapid as expressed below:

```
/* TCP-FS with Rapid */

class RapidTcpFsAgent : public RapidTcpAgent, public
TcpFsAgent {
public:
RapidTcpFsAgent() : RapidTcpAgent(), TcpFsAgent() {}

      /* helper functions */

virtual void output_helper(Packet* pkt)
{TcpFsAgent::output_helper(pkt);}
virtual void recv_helper(Packet* pkt)
{TcpFsAgent::recv_helper(pkt);}
virtual void send_helper(intmaxburst)
{TcpFsAgent::send_helper(maxburst);}
virtual void send_idle_helper()
{TcpFsAgent::send_idle_helper();}
virtual void recv_newack_helper(Packet* pkt)
{TcpFsAgent::recv_newack_helper(pkt);}

virtual void set_rtx_timer()
{TcpFsAgent::set_rtx_timer();}
virtual void cancel_rtx_timer()
{TcpFsAgent::cancel_rtx_timer();}
virtual void
cancel_timers(){TcpFsAgent::cancel_timers();}
virtual void timeout_nonrtx(inttno)
{TcpFsAgent::timeout_nonrtx(tno);}
virtual void timeout_nonrtx_helper(inttno);
};
```

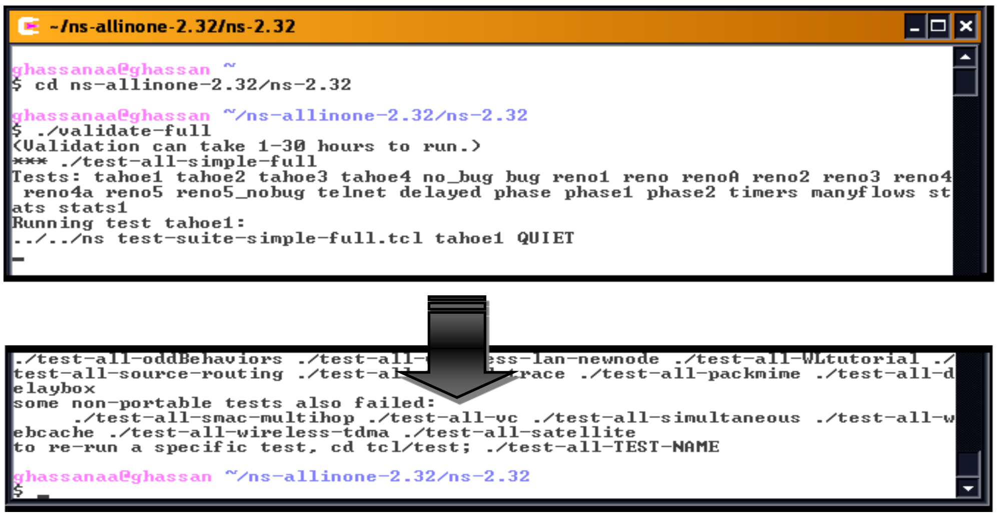
6. The last stage is to develop a copy of tcp-Reno.cc files and TCP- Reno.h and rename these files to become tcp-Rapid.cc and tip-Rapid.h respectively. TCP files, tcp-Rapid.h distinguishes the header file where they will be fixing agent guidance and the necessary timing functions that perform rapid TCP protocol. In contrast, the file tcp-Rapid.cc in this file really is performed every timing, TCL hooks, agent guidance.

After theconfiguration and the ratification of NS-2 simulator to be able to identify different protocols. Actually, the representation of new TCP based on the standard ofTCP- Reno and carries the same architecture and the same control mechanism in the congestion state. But that TCP will bearranged to edit control in the traditional congestion based on AIMD (1, ½), which was established with proposed slow start at the beginning of an algorithm and it is installed in tcp.cc file instead of exponential algorithm slow start. And byusing many of the commands to set up and manipulate TCP flows rapidly in the simulation, which could soon explain the configuration parameters and how to assign Fast TCP connection:

```
set tcp [new Agent/TCP/Rapid] ;     # create tcp agent
$ns_ attach-agent $node_(s1) $tcp ; # bind src to node
$tcp set fid_ 0 ;# set flow ID field
set ftp [new Application/FTP] ;# create ftp traffic
$ftp attach-agent $tcp ;# bind ftp traffic
                                        # to tcp agent
set sink [new Agent/TCPSink] ;# create tcpsink agent

$ns_ attach-agent $node_(k1) $sink ;# bind sink to node
$sink set fid_ 0 ;# set flow ID field
$ns_ connect $ftp $sink ;# active connection
                                        #src to sink
$ns_ at $start-time "$ftp start" ;# start ftp flow
```

As a result, currently the rapid TCP agent is very similar to TCP Tahoe agent extent, but it includes a rapid recovery mode where they are inflatable recent cwnd via the sum of DUPACKs that TCP source expected it before receiving the next ACK. Moreover, the new TCP agent is not due to TCP slow start by re-setting move fast. Instead, it reduces cwnd sets to half, and its current size sets ssthresh to match this value. The last stage inconsidering the new TCP in NS-2 is by configuring and validating, to guarantee that, the new TCP becomes well-matched and identifiable byNS-2 modules and components. This process performed by three spilt commands: ./configure, make, and ./validate-full (in NS-2.34 it should be used ./validate instead of validate-full). Figure 2illustrates the screenshot of the finished authentication procedure.



```
ghassanaa@ghassan ~
$ cd ns-allinone-2.32/ns-2.32

ghassanaa@ghassan ~/ns-allinone-2.32/ns-2.32
$ ./validate-full
(Validation can take 1-30 hours to run.)
*** ./test-all-simple-full
Tests: tahoe1 tahoe2 tahoe3 tahoe4 no_bug bug reno1 reno renoA reno2 reno3 reno4
 reno4a reno5 reno5_nobug telnet delayed phase phase1 phase2 timers manyflows st
ats stats1
Running test tahoe1:
../../ns test-suite-simple-full.tcl tahoe1 QUIET
_
```

```
./test-all-oddBehaviors ./test-all-       ess-lan-newnode ./test-all-WLtutorial ./
test-all-source-routing ./test-al          race ./test-all-packmime ./test-all-d
elaybox
some non-portable tests also failed:
      ./test-all-smac-multihop ./test-all-vc ./test-all-simultaneous ./test-all-w
ebcache ./test-all-wireless-tdma ./test-all-satellite
to re-run a specific test, cd tcl/test; ./test-all-TEST-NAME

ghassanaa@ghassan ~/ns-allinone-2.32/ns-2.32
$ _
```

Figure 2. Validation of NS-2 to ensure the accumulating the new TCP

Also, a simple script is used to test the functionality of the proposed TCP with simple topology, to demonstrate that TCP is prepared to be investigated and ran over wireless and high speed links model. Figure 3 shows the typical congestion window of TCP over simple network model.

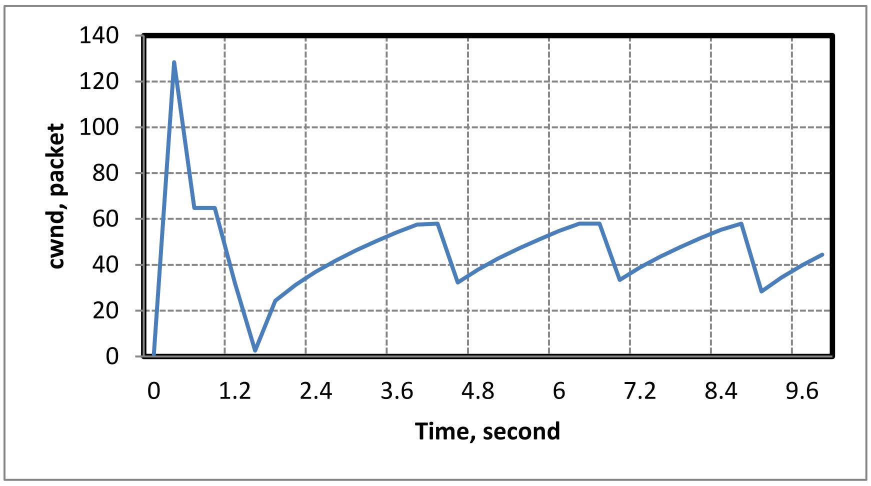


Figure 3. The standard congestion window of TCP Rapid

6. Conclusion

In this article, a new TCP called TCP-Rapid including an innovative control algorithm to avoid congestion furthermore to a slow start algorithm. Rapid TCP holds the general theories of TCP-Reno, but it is grounded on new AIMD algorithm with a new relationship between the pair parameters a and b. In addition, the association between these pairs can provide additional number of virtual flows of Rapid TCP instead of one flow inReno, which allows rapid to provide a high level of access to productive to k provided productive times of one flow in Reno. Other large capacity of RAPID is to provide a new algorithm to control congestion to estimate the number of optimal dynamic flows away from the setup by the user. This paper is also involved in the creation of rapid agent in NS-2 models designed to identify and rapid TCP and configure the NS-2 platform. It has a fast TCP thatalso includes an innovative scheme to slow the rapid start to help TCP to start faster through the high speed networks and also to postpone the congestion state as much as possible.

References

[1] Simulator, N., ns-2. 1989.
[2] Jacobson, V. Congestion avoidance and control. 1988: ACM.
[3] Ghassan A Abed, Mahamod Ismail, JumariKasmiran. Improvement of TCP Congestion Window over LTE-Advanced Networks. Faculty of Engineering and Built Environment the National University of Malaysia. *International Journal of Advanced Research in Computer and Communication Engineering*. 2012; 1(4): 2012.
[4] Liu, Y, et al. Developing Slow Start over Wireless Networks. 2008: *IEEE*.
[5] K Saniee. A Simple Expression for Multivariate Lagrange Interpolation. *New Providence High School, New Providence*. 2007; NJ 07974.
[6] M Gasca, T Sauer. Polynomial interpolation in several variables. *Advances in Computational Mathematics*. 2000; 12: 377-410.
[7] Ghassan A Abed, Mahamod Ismail, KasmiranJumari. *Modeling and Performance Evaluation of LTE Networks with Different TCP Variants*. Proc. of World Academy of Science, Eng. and Tech, 2011.
[8] A Vizzarri. *Analysis of VoLTE end-to-end quality of service using OPNET*. Department of Engineering Enterprise University of Rome Tor Vergata. 8th European modeling Symposium on Mathematical Modelling and Computer Simulation. 2014.