academicJournals

Vol. 9(10), pp. 452-457, 30 May, 2014
DOI: 10.5897/SRE2013.5790
Article Number: 0590E3044941
ISSN 1992-2248

http://www.academicjournals.org/SRE

Scientific Research and Essays

Full Length Research Paper

Generation and simulation of new transmission control protocol (TCP) agent over network simulator 2 (NS-2) platforms

Bayan M. Sabbar

College of Information Engineering, Al-Nahrain University, Baghdad, Iraq.

Received 31 December, 2013; Accepted 2 April, 2014

Transmission Control Protocol (TCP) is the most popular protocol used over the wired and wireless networks, and it still has a practical problem where the congestion control mechanism does not permit the data stream to get complete bandwidth over the existing network links. To solve this problem, many TCP protocols have been introduced with high speed performance. The work provided in this article is based on using the Network Simulator 2 (NS-2) to implement a new proposed TCP called "Sumer" TCP. The proposed TCP is based on the same characterizations of Reno TCP, such as congestion control, sliding window and other Reno mechanisms and features.The process of creating new TCP agent over the platforms of NS-2 requires a lot of modifications, files generation, and agent identification to make the new agent recognizable by NS-2 resources. The proposed TCP is developed for scientist and researchers to be easy for them to add the new mechanisms such as slow-start and congestion avoidance with other extra features.

Key words: Transmission Control Protocol (TCP), Sumer TCP, Reno, congestion control, congestion window, NS-2

INTRODUCTION

Transmission Control Protocol (TCP) is a basic communication language, and a connection oriented protocol tied with transport layer that consists of collection of rules and procedures to control communication between links (Abed et al., 2011a). There are many TCP variants that modified and developed respectively with the communications needs. Most TCP present forms include set of algorithms built to control the congestion in critical links of network while maintaining the network throughput. In current years, TCP has been faced with the fast growth in internet in parallel with the increasing demand to transfer the media on high speed links supported TCP. In the last years, computer networks and mobile cellular systems have qualified incredible evolution and a lot of computers and other user equipment's become linked together with most mutual protocol stack used being TCP. Currently, it is hard to recognize the congestion control mechanisms that are applied by different engines in Internet. One more imperative problem is the manner that these mechanisms are employed in diverse operating systems (Abed et al., 2011b). The greatest universal transport protocol involved is the TCP and in the original

Email: iraqi_man73@yahoo.com

accomplishment of TCP, a very small number of variants were done to minimalize the congestion in network path. Additionally, TCP employment uses accumulative confident acknowledgements and the expiration of a retransmission timer to afford reliability based on a modest go-back-n model. TCP implements various techniques that use proficiently the network resources by approximating the conditions and the characteristics of network. Also, TCP is built on the concept of self-clocking technique where this concept based on several characteristics. TCP has become the key factor in manipulating the behavior and performance of the networks. The TCP congestion control plays a vital role in controlling the applications that request the services over various networks. Furthermore, the congestion control provides the amount of traffics that can be inserted into the network, where it overrides the behavior and the performance of the communications processes.

TCP MODELING IN NS-2

The network simulator NS-2 is a free-access and object-oriented with discrete-event network simulator (NS-2, 1989). NS-2 provides a structure for constructing a network prototype and identifies data as input parameters, analyzing data output and giving outcomes and results. Two main reasons for the wide impact of NS-2 are as follows, the first is because it is free, where that fits researchers in laboratories and universities, and the second reason is the huge range of network modules and objects that can be implemented by NS-2 (Olsén, 2003). This article is used to provide user interfacing that permits the specified input of the model (Tcl scripts) to be executed. Mostly, the elements of any network topology in NS-2 are established as classes in object oriented style. For TCP modeling, NS-2 offers significant support for TCP simulating, modeling, queuing algorithms, routing algorithms, and multicast protocols. Modeling of TCP in NS-2 was initially based on the source code of the BSD kernel (Berkeley Software Distribution) in the 1990s (Wei and Cao, 2006). Later, the TCP modules in NS-2 have extremely assisted the research teams and groups to evaluate and investigate the behavior of TCP where that led to the expansion in developing many congestion control techniques. Two categories of TCP are available in NS-2. The first category is a one way TCP, where it uses objects with several classes on sides, sender and receiver. On the TCP sender side, some available classes are provided for TCP Tahoe, Newreno, Reno, Vegas Sack, and Fack. While in the side of receiver, the classes available for TCP are both without delayed and with selective acknowledgements. Furthermore, other subclasses can be derived from these provided classes to apply the required modifications to the standard congestion control mechanisms. The second category includes two ways TCP, where the TCP uses objects with the same class on sender and receiver sides. The NS-2 is written using C++ as a programming language, with an OTcl interpreter shell (Tcl is a script language with Object-Oriented extensions Tcl: Tool Command Language). Figure 1 show the code structure of TCP Linux in NS-2 where TCP Linux is an experimented TCP created by Wieand Cao (2006).

The whole modules include four parts, corresponding to the four white blocks in Figure 1. The yellow blocks are from outside source codes such as NS-2 or Linux (Wie and Cao, 2006). TCPLinuxAgent (in tcp-linux.h and tcp-linux.cc): this is the main component which loosely follows the Linux implementation in packet receiving, ack processing and congestion control. ScoreBoard1 (in scoreboard1.h and scoreboard1.cc): this is a new packet SACK/Loss/Retransmission control module which combines Scoreboard-rq in NS-2 and Linux's ACK/SACK processes. It loosely follows the steps in tcp_clean_rtx_queue and tcp_sacktag_write_queue in tcp_input.c in Linux. The interface between NS-2 and Linux (in linux/ns-linux-util.h and .cc): this part redefines the data structure in Linux TCP and provides interfaces between the NS-2's C++ code and Linux's C code. Shortcuts for other Linux system calls (in linux/ns-linux-c.h and .c): this part redefines many system calls in Linux (to void) and allows Linux source code to be compiled with very minor changes.

In Figure 2 we can see that the NS-2 design uses a model named shared object design where this means that the NS-2 system is based on programming in two languages and with these two languages there is a corresponding hierarchy of the network objects, but the object in one are open to the other and also there is an object accessible to one portion of the system where this is basically for an efficiency purpose. NS-2 uses C++ to write and compile the network components in the data path to reduce the packet and the required time for processing. The objects compiled by the NS-2 system are made to exist to an OTcl interpreter over an article linkage.

This linkage generates an equivalent OTcl object for each C++ object and creates the configurable variables identified by C++ objects to affect as an associated variable and function to the corresponding article objects. In this manner, the controlling of C++ objects agrees to OTcl which enables changing the linked variables of C++ from a script of Tcl. Also, it is possible to add variables and functions as C++ linked of OTcl object. Certainly, some of C++ objects that do not need control during the simulation are internally used by other objects that do not require to be connected to OTcl. Figure 3 illustrates the linkage between C++ and OTcl (Wang, 2004).

The user (not necessarily the developer of NS) can be assumed to be standing in left bottom corner, designing and executing the simulations in Tcl by using the simulated objects in the library of OTcl. Then movement from this point to the right top corner which gives more

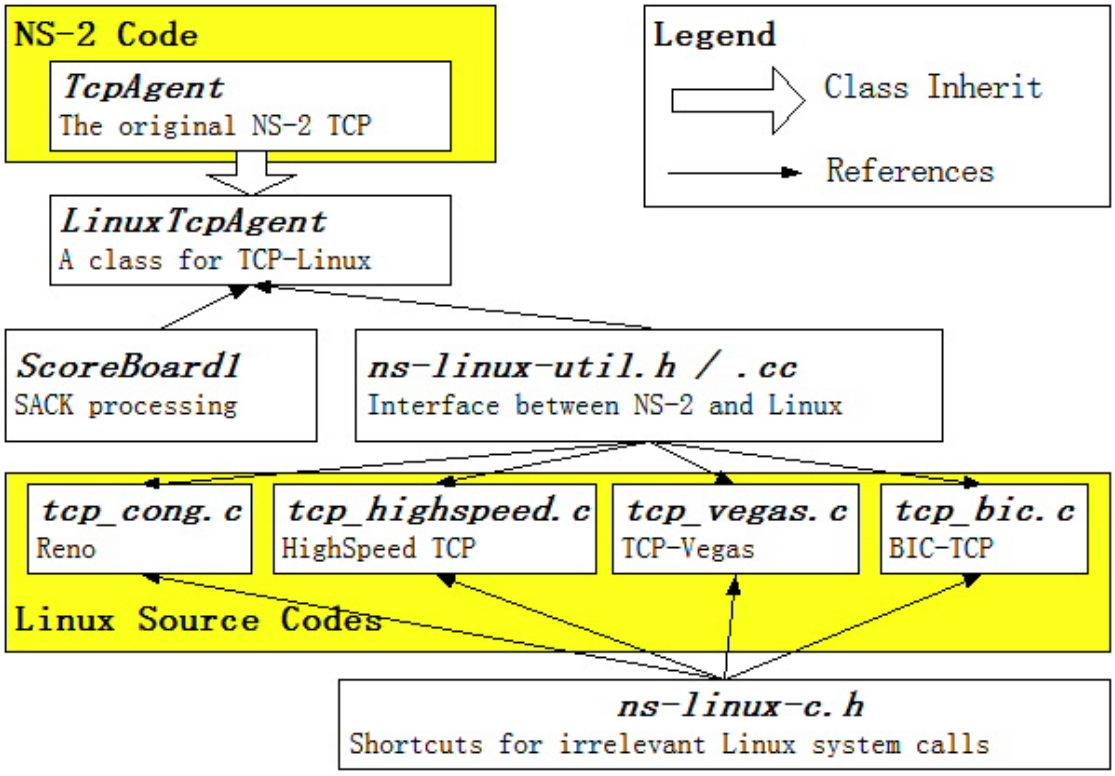

Figure 1. Code structure of TCP Linux in NS-2.

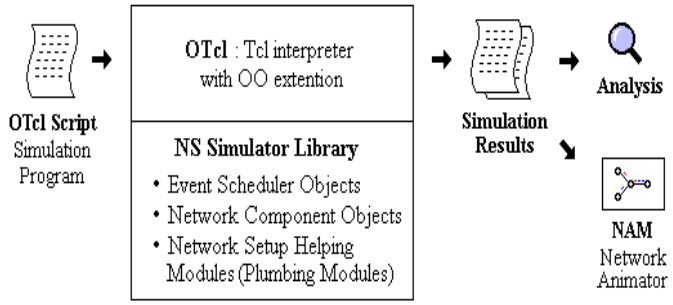

Figure 2. Simplified user's view of NS-2.

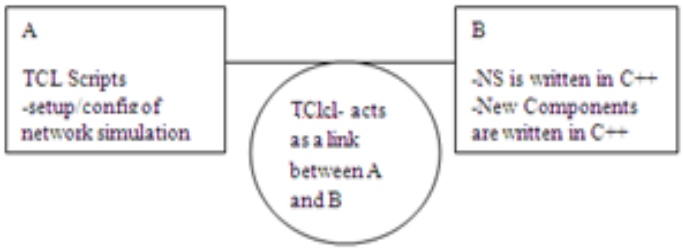

Figure 3. TclCl provides the linkage between C++ and OTcl.

understanding and knowledge of NS-2 as a whole is wanted. The event scheduler and a lot of network components are executed using C++ language existing as OTcl over a linked article, and is applied using Tcl with classes (tclcl) as a Tcl/C++ interface.

SPECIFICATIONS OF SUMER TCP

The state diagram shown in Figure 4 consists of many phases; however, all these phases represent an integrated and enhanced congestion control algorithm to control the size of congestion window (cwnd) in professional approach with high throughput. As explained, Sumer TCP is an improved Reno with high performance congestion control. Every time when three DUPACK's receive ACK's that means the segment is already lost and the algorithm retransmit this segment again and enter fast recovery mode.

Also, it sets the slow-start threshold (ssthresh) to the half of the current size of cwnd and set cwnd to become cwnd-cwnd*(2/(3k+1)) where k has already been estimated. On one side, for every DUPACK received, cwnd increases by k/cwnd and when the increasing of cwnd has exceeded the amount of segments in the network pipe there will be transmission of new segment delays. In slow-start phase, the new congestion control uses the duplicated increment with quadratic interpolation to achieve faster increase. The effect of this technique should be used in initial starting up (when the connection

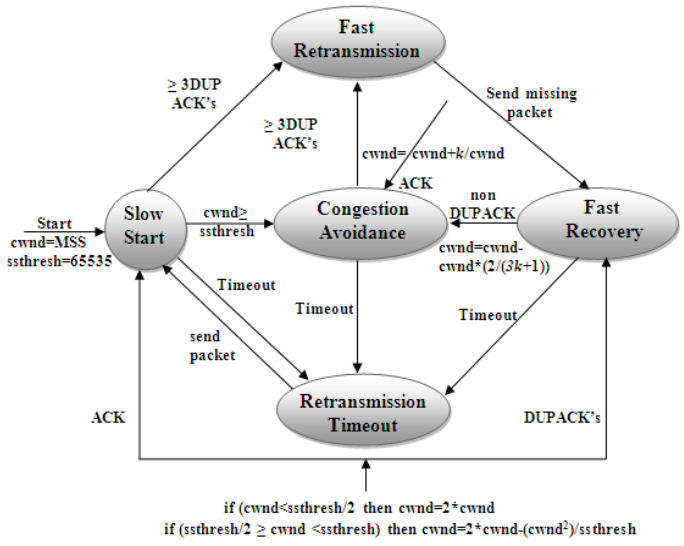

Figure 4. State transition diagram of proposed congestion control mechanism.

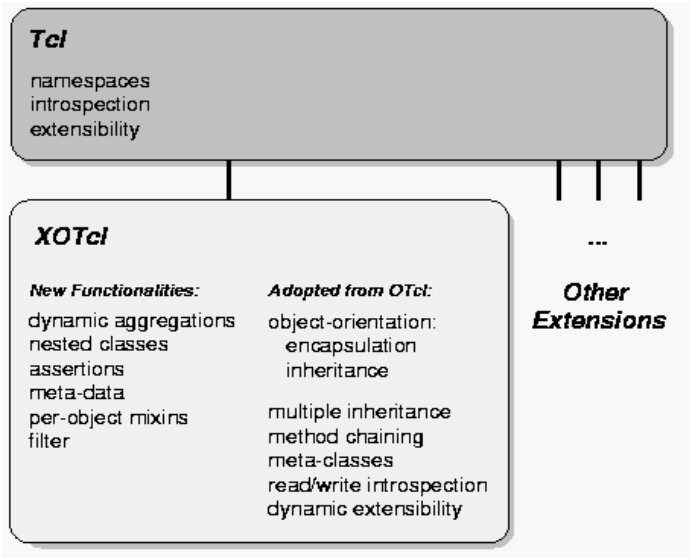

Figure 5. Language extensions of XOTcl.

is established) and after fast recovery mode to obtain faster growth and then delay the congestion avoidance phase as much as possible. In first start up, the cwnd is set to be equal to the MSS (the default value is 1460 Bytes) while the ssthresh is set to be 65535 Bytes, but after the initialization, the cwnd begins estimating according to improved AIMD and the ssthresh according to the network status.

GENERATING SUMER TCP

Many TCP protocol implementations are added to NS-2 such as *Agent/TCP/Reno*, *Agent/TCP/Newreno, Agent/TCP/Sack1, Agent/TCP/Vegas, Agent/TCP/Fack, and Agent/TCP* where this one mentions to TCP Tahoe. The main goal here is to create a new TCP agent for Sumer TCP, which is to be identified by NS-2 components and also include the proposed congestion control algorithm. The big challenge in modeling, developing and simulating the networks models and protocols of the researchers, students, and developers are using NS-2 due to the low-level programing language used in this simulator. In this research, the released NS-2.34 is used and this version is installed over Windows XP using Cygwin, where Cygwin provides a Linux-like environment under Windows.

As shown in Figure. 5, the modeling of new module in NS-2 should be constructed with sides, C++ class and OTcl class. The class Tcl Object represents the base class of the most of the other classes in the compiled and interpreted hierarchies.

All objects in the class Tcl Object are generated by the users from inside the interpreter while the equivalent shadow object is generated in the compiled side of the hierarchy and these two objects are narrowly accompanied with each other. The other important class called Tcl Class, represents a pure virtual class in the NS-2 system.

Sumer TCP is built on the same concept of TCP Reno with modification in congestion control, and then the modeling and implementation of Sumer TCP will use the source files of Reno and modify each file separately then use the modified files to create Sumer TCP as a separated TCP agent. For Sumer TCP, it is considered the class becomes SumerTcp Class where it is derived from Tcl Class and is associated with the class SumerTcp Agent. The compiled class hierarchy of SumerTcp Agent is derived from Tcp Agent and that in turn derived from the Agent that in turn roughly derived from Tcl Object.

SumerTcpClass is defined as:

```
static class SumerTcpClass: public TclClass {
public:
SumerTcpClass() : TclClass("Agent/TCP/Sumer") {}
TclObject* create(intargc, const char*const* argv) {
return (new SumerTcpAgent());
        }
        } class_Sumer;
```

NS-2 will execute the constructor of SumerTcpClass for the static variable class_Sumer when it is first initiated and this setup is both the appropriate approach and interpreted class hierarchy.

In addition, the constructor states the interpreted class clearly as *Agent/TCP/Sumer* and this identifies the interpreted class hierarchy implicitly. The convention in NS-2 is to use the slash character “/”as a separator and for any assumed class A/B/C, the A/B/C class represents

a subclass of A/B where that is a subclass of A, where, A itself is a subclass of TclObject. In SumerTcpClass case, the constructer of TclClass builds three classes, *Agent/TCP/Sumer* subclass of *Agent/TCP* subclass of *Agent* subclass of Tcl Object. The created class is associated with the class SumerTcpAgent and it creates new objects in the associated class. The SumerTcpClass creates method returns TclObjects in the class SumerTcpAgent and if the user identifies new *Agent/TCP/Sumer,* the routine SumerTcpClass is invoked.

MODIFICATIONS IN NS-2

To accept Sumer TCP as a new protocol over the NS-2 platform, many source files are required to create (or modify) to make the RTCP recognizable and ready to merge the new congestion control algorithm. Unfortunately, the procedures to add RTCP in NS-2 files is very sensitive and complex, because there is no professional documentation for these routines and the other risk such as the NS-2 does not include full help in compiling operation (all required modifications done using C++), so when the developers face an error, they should revise all the performed steps. The modified files involved here are based on the version 2.3x of NS; then the proposed TCP termed Sumer is assumed to be experimented over NS-2.3x series. The first modification is applied on ns-compact.tcl file in the location */ns-allinone-2.3x/ns-2.3x/tcl/lib/* as shown below:

By adding:
$self map_ns_defaultsns_Sumertcp

Then adding:
Agent/TCP/Sumer
TclObject set varMap_(rampdown) rampdown_
TclObject set varMap_(ss-div4) ss-div4_

Then adding:
setclassMap_(tcp-Sumer)
Agent/TCP/Sumer
set classMap_(Sumertcp)
Agent/TCP/Sumer

Then, the file Makefile.in which locates in: */ns-allinone-2.3x/ns-2.3x/Makefile.in* needs to add a single line in the same groups with the other TCP variants:tcp/tcp-Sumer.o
Other single line to the file ns_tcl.cc which locates in: */ns-allinone-2.3x/ns-2.3x/gen/ns_tcl.cc*:
$self map_ns_defaultsns_Sumertcp\n\

Two short lines should be added to FILES in the main NS folder ns-2.3x:

tcp/tcp-Sumer.cc
tcp/tcp-Sumer.h

In the same file, the next four lines are added as shown below:

tcl/test/test-output-tcpVariants/fourdrops_Sumer.gz
tcl/test/test-output-tcpVariants/onedrop_Sumer.gz
tcl/test/test-output-tcpVariants/threedrops_Sumer.gz
tcl/test/test-output-tcpVariants/twodrops_Sumer.gz

The last step is to get a copy of the files tcp-Reno.cc and tcp-Reno.h and rename these two files to become tcp-Sumer.cc and tcp-Sumer.h respectively. The file tcp-Sumer.h characterizes the header file where will be defined the routing agent and all necessary timers which performs the functionality of the Sumer TCP protocol. To validate the generating of Sumer TCP over NS-2, we experimented the new agent by drawing the congestion window of packet transmission in simple network topology. The first test was based on plotting the congestion window without congestion event as shown in Figure 6 where we used 60 packets as a window size and 20 s as simulation period.

In this figure we can note the typical congestion window of Sumer TCP and we can observe the new behavior of Sumer TCP without congestion event in the simulation scenario. The other test of Sumer TCP proceeded with simple congestion event by adding packet loss to the link and proposed topology by reducing the bandwidth of the network bottleneck as shown in Figure 7. The last test proceeded to validate the congestion control mechanism behavior when the network suffers from congestion events by adding some cross links and increasing the packet loss. In this test the window size was 20 packets and the simulation period was 10 s.

CONCLUSIONS

In this paper, a new TCP called Sumer was created and is based and carried out on the general concepts and features of TCP Reno. The improved TCP used new congestion control mechanism in enhancing the window behavior in slow-start and congestion avoidance phases. This paper had also involved in creating the agent of Sumer in NS-2 modeler and identifying and configuring Sumer TCP over the NS-2 platform. The new TCP agent was generated by adding many subroutines, algorithms, and functions to make the new agent readable by the NS-2 compiler. Furthermore, the TCP modeling is required to add many source files and headers to build the structure of the new TCP where the new files and headers are programmed using C++ language. The pseudo code of all files, routines, and subroutines are illustrated in this article.

FUTURE WORK

Further researches on this article will emphasize modification

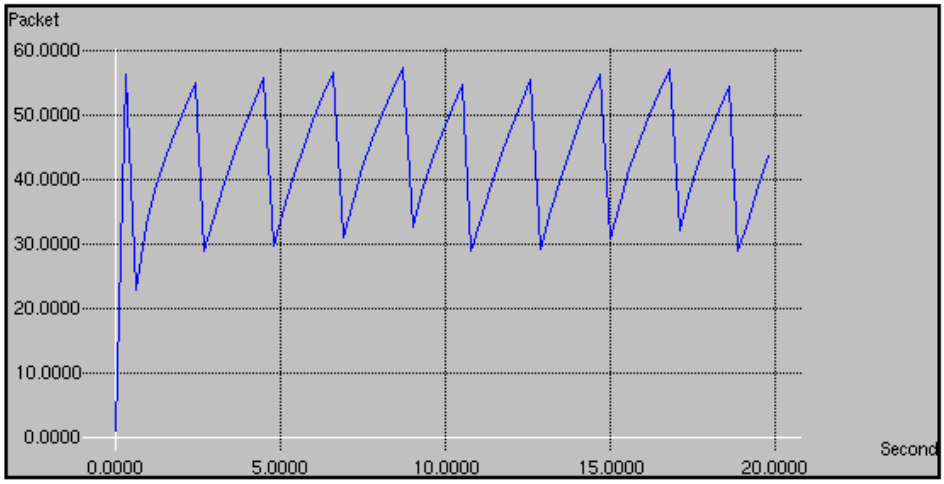

Figure 6. Congestion window of Sumer TCP without network congestion.

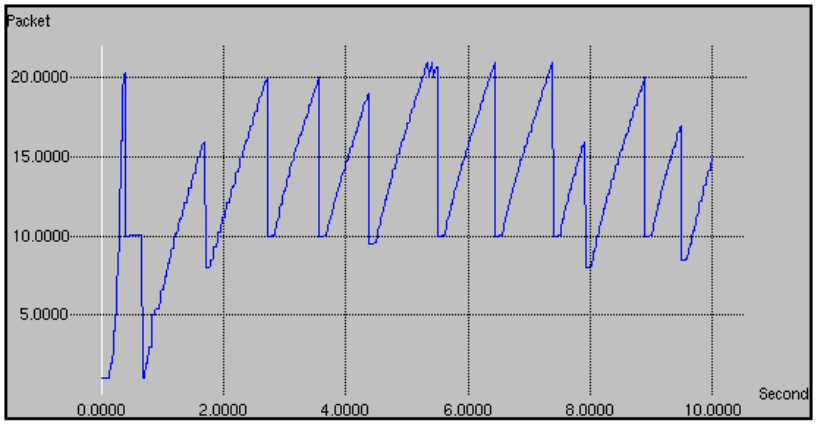

Figure 7. Congestion window of Sumer TCP with network congestion.

of the congestion control mechanism of Sumer TCP to give the new agent some private and useful features.

Conflict of Interests

The author(s) have not declared any conflict of interests.

REFERENCES

Abed GA, Ismail M, Jumari K (2011a). Architecture And Functional Structure Of Transmission Control Protocol Over Various Networks Applications. J. Theor. Appl. Inf. Technol. 34(1).

Abed GA, Ismail M, Jumari K (2011b). A Comparison And Analysis Of Congestion Window For Hs-Tcp, Full-Tcp, And Tcp-Linux In Long Term Evolution System Model.2011 IEEE Conference on Open System (ICOS). pp. 358-362. http://dx.doi.org/10.1109/ICOS.2011.6079287

Olsén J (2003). Stochastic Modeling And Simulation Of The TCP Protocol. MatematiskaInstitutionen, Univ.NS-2 (1989). Network Simulator.

Wang J (2004). NS-2 Tutorial. Multimedia Networking Group, The Department Of Computer Science, UVA.

Wei DX, Cao P (2006). Ns-2 Tcp-Linux: An Ns-2 Tcp Implementation With Congestion Control Algorithms From Linux.