ORIENTAL JOURNAL OF
COMPUTER SCIENCE & TECHNOLOGY
An International Open Free Access, Peer Reviewed Research Journal
Published By: **Oriental Scientific Publishing Co., India.**
www.computerscijournal.org

ISSN: 0974-6471
September 2013,
Vol. 6, No. (3):
Pgs. 363-369

Adding TCP-Variants to NS-2

AHMED JAWAD KADHIM

Ministry of Education, General Directorate for Education Qadisiyah, Iraq.

(Received: August 21, 2013; Accepted: August 30, 2013)

ABSTRACT

The network simulator (NS-2) is very important to simulate the network types such as mobile ad-hoc network efficiently and easily by providing the environment of this network exactly. This simulator helps the researcher in the last years to introduce Their researches without need to the expensive requirements to build the network in real time. This simulator uses file (called *cbegen.tcl*) to generate the traffic between nodes of the network at random time according to uniform distribution. The original *cbrgen.tcl* file provides two types of traffics that are CBR with UDP and FTP with TCP. The purpose of this paper is to add the one-way TCP and two-way TCP variants to NS-2. Also, this paper made these variants operate with many types of sink such as TCPSink, TCPSink/DelAck, TCPSink/Sack1, and TCPSink/Sack1/DelAck. This addition make it usable for anyone that wants to study the behaviour of these variants and its effects on the network.

Key word: NS-2, one-way TCP, two-way TCP MANET.

INTRODUCTION

Network simulator (NS-2) is a good tool for the purpose of network simulation. NS-2 can simulate many types of network like LAN and WPAN depending on script written by the user. NS-2 depends on two programming languages C++ and OTCL. C++ used to make the execution very faster while OTCL used to create network environment and its important occur in modification of the network environment in quickly manner[1]. The traffic between the source and destination nodes can be ~~is~~ generated by using *cbrgen.tcl* file. This file can generate only CBR with UDP and FTP with Tahoe TCP[2]. UDP support an unreliable connectionless between two hosts in the network[3]. TCP is responsible for a reliable and connection-oriented communication because the connection is established prior to transmitting data .In TCP there is a guarantee that the data is being transmitted to the destination[4]. In TCP results long delay because there are many requests for the lost packets[5]. There are many types of TCP that are one-way and two-way as a source. One-way TCP are Tahoe, Reno, Newreno, Sack1, Fack, Vegas, and Linux. Two-way TCP is FullTcp. As a sink also there are many types that are TCPSink, TCPSink/DelAck, TCPSink/Sack1, and TCPSink/Sack1/DelAck[6].

Related Work

Saad Talib Hasson and et al. at 2012 add exponential on/off, Pareto on/off, and Telnet traffic generators to the *cbrgen.tcl* file in NS-2, as well as made the time of the traffic generator distributed between 0 and the simulation time instead of 0 and 180. They concluded that can be generate many traffics in good manner through the simulation time and prevent the traffics that can be schedule out the range of the simulation period[7].

This paper also helps in modifying the *cbrgen.tcl* file but this addition associated with other types of the traffic generators (one-way and two-way TCP) that can be work with different types of sink like TCPSink, TCPSink/DelAck, TCPSink/Sack1, and TCPSink/Sack1/DelAck.

Modified File

This section shows the code of the modified file that can be used in NS-2 easily to build the traffic generators between the source and destination nodes in the mobile ad-hoc networks.

To make NS-2 accept this modification, the user must be enter to the following directory *ns-allinone-2.34/ns-2.34/indep-utils/cmu-scen-gen*, and then open *cbrgen.tcl* file and delete original code and put in it the following code.

```
set opt(nn)                0            ;    #
Number of Nodes
set opt(seed)              0.0
set opt(mc)                0
set opt(pktsize)           512
set opt(traffic)           “”
set opt(rate)              0
set opt(interval)          0.0          ;# inverse of
rate
set opt(type)              “”
set opt(t)                 0
set opt(destination)       “”
#
=============================================

proc usage {} {
   global argv0
puts “\nusage: $argv0 \[-type UDP|TCP|TCP/
Vegas|TCP/Linux|TCP/Fack|TCP/Newreno|TCP/
Reno|TCP/Sack1|TCP/FullTcp\] \[-traffic
exp|pareto|cbr|FTP|Telnet\] \[-destination
TCPSink|TCPSink/DelAck|TCPSink/
Sack1|TCPSink/Sack1/DelAck\] \[-nn nodes\] \[-
seed seed\] \[-mc connections\] \[-rate rate\] \[-t t\]\n”
   }
#=============================================

proc getopt {argc argv} {
        global opt
        lappend optlist type traffic destination nn
seed mc rate t

        for {set i 0} {$i < $argc} {incr i} {
                set arg [lindex $argv $i]
                if {[string range $arg 0 0] != “-”}
continue

                set name [string range $arg 1
end]
                set opt($name) [lindex $argv
[expr $i+1]]
        }
}
#=============================================

if {$argc !=16} {
    puts “error”
    usage
exit 1
}

getopt $argc $argv

#
=============================================
proc create-udp-all-connection { src dst } {

    global rng cbr_cnt opt opt\(type\) traffic nn t
        set stime [$rng uniform 0.0 $opt(t)]

        puts “#\n# $src connecting to $dst at time
$stime\n#”
        puts “set udp_($cbr_cnt) \[new Agent/
UDP\]”
        puts “\$ns_ attach-agent \$node_($src)
\$udp_($cbr_cnt)”
        puts “set null_($cbr_cnt) \[new Agent/
Null\]”
        puts “\$ns_ attach-agent \$node_($dst)
\$null_($cbr_cnt)”
```

```
        if { $opt(traffic) == "cbr" } {
        puts "set cbr_($cbr_cnt) \[new Application/
Traffic/CBR\]"
        puts "\$cbr_($cbr_cnt) set packetSize_
$opt(pktsize)"
        puts "\$cbr_($cbr_cnt) set interval_
$opt(interval)"
        puts "\$cbr_($cbr_cnt) set random_ 1"
        puts "\$cbr_($cbr_cnt) set maxpkts_
10000"
       puts "\$cbr_($cbr_cnt) attach-agent
\$udp_($cbr_cnt)"
        puts "\$ns_ connect \$udp_($cbr_cnt)
\$null_($cbr_cnt)"
        puts "\$ns_ at $stime \"\$cbr_($cbr_cnt)
start\""

        } elseif { $opt(traffic) == "exp" } {
        puts "set exp_($cbr_cnt) \[new
Application/Traffic/Exponential\]"
        puts "\$exp_($cbr_cnt) set packetSize_
$opt(pktsize)"
        puts "\$exp_($cbr_cnt) set rate_
$opt(rate)\kb"
        puts "\$exp_($cbr_cnt) set burst_time_
0.5"
        puts "\$exp_($cbr_cnt) set idle_time_ 0.5"
        puts "\$exp_($cbr_cnt) attach-agent
\$udp_($cbr_cnt)"
        puts "\$ns_ connect \$udp_($cbr_cnt)
\$null_($cbr_cnt)"
        puts "\$ns_ at $stime \"\$exp_($cbr_cnt)
start\""

        } else {
        puts "set pareto_($cbr_cnt) \[new
Application/Traffic/Pareto\]"
        puts "\$pareto_($cbr_cnt) set
packetSize_ $opt(pktsize)"
       puts "\$pareto_($cbr_cnt) set rate_
$opt(rate)\kb"
        puts "\$pareto_($cbr_cnt) set burst_time_
0.5"
        puts "\$pareto_($cbr_cnt) set idle_time_
0.5"
        puts "\$pareto_($cbr_cnt) set shape_ 1.5"
        puts "\$pareto_($cbr_cnt) attach-agent
\$udp_($cbr_cnt)"
        puts "\$ns_ connect \$udp_($cbr_cnt)
\$null_($cbr_cnt)"
        puts "\$ns_ at $stime
\"\$pareto_($cbr_cnt) start\""

}
        incr cbr_cnt
}

#=============================================
proc create-tcp-all-connection { src dst } {
        global rng cbr_cnt opt opt\(type\) traffic
destination nn t
        set stime [$rng uniform 0.0 $opt(t)]

      puts "#\n# $src connecting to $dst at time
$stime\n#"
        puts "set tcp_($cbr_cnt) \[\$ns_ create-
connection \
       $opt(type) \$node_($src) $opt(destination)
\$node_($dst) 0\]";
        puts "\$tcp_($cbr_cnt) set window_ 32"
        puts "\$tcp_($cbr_cnt) set packetSize_
$opt(pktsize)"
        puts "set ftp_($cbr_cnt) \[\$tcp_($cbr_cnt)
attach-source $opt(traffic)\]"
        puts "\$ns_ at $stime \"\$ftp_($cbr_cnt)
start\""
        incr cbr_cnt

}

#=============================================

if { $opt(nn) == 0 || $opt(seed) == 0.0 || $opt(mc) ==
0 || $opt(rate) == 0 || $opt(traffic) == "" ||
$opt(destination)== "" || $opt(type) == "" } {

        usage
        exit
  }
  set opt(interval) [expr 1 / $opt(rate)]
  if { $opt(interval) <= 0.0 } {
        puts "\ninvalid sending rate $opt(rate)\n"
        exit
  }

puts "#\n# nodes: $opt(nn), max conn: $opt(mc),
send rate: $opt(interval), seed: $opt(seed)\n#"

set rng [new RNG]
$rng seed $opt(seed)
```

Fig. 1: Number of the dropped packets for each TCP type

Fig. 2: The throughput for each TCP type

Fig. 3: Average jitter for each TCP type

Fig. 4: Normalize routing load for each TCP type

```
set u [new RandomVariable/Uniform]
$u set min_ 0
$u set max_ $opt(t)
$u use-rng $rng

set cbr_cnt 0
set src_cnt 0

for {set i 0} {$i < $opt(nn) } {incr i} {

        set x [$u value]
    if {$x < 50} {continue;}

        incr src_cnt

        set dst [expr ($i+1) % [expr $opt(nn) + 1] ]


        if { $opt(type) == "UDP" } {

        create-udp-all-connection $i $dst

        } else {
        create-tcp-all-connection $i $dst

}

if { $cbr_cnt == $opt(mc) } {
                break
        }

   if {$x < 75} {continue;}

        set dst [expr ($i+2) % [expr $opt(nn) + 1] ]

     if { $opt(type) == "UDP" } {

                create-udp-all-connection $i
$dst

        } else {

                create-tcp-all-connection $i $dst

}

        if { $cbr_cnt == $opt(mc) } {
                break
        }
}
puts "#\n#Total sources/connections: $src_cnt/
$cbr_cnt\n#"

#=====================================================
```

After putting the above code in the *cbrgen.tcl* file, the user must be writing the following instruction to generate the traffics that are one-way and two-way TCP variants with many types of sink easily.

ns cbrgen.tcl –type agent_type –traffic traffic_type -destination destination_type –nn number -seed number -mc number -rate number –time simulation_time>file_name

Where:
agent_type : either UDP, TCP, TCP/Vegas, TCP/Linux, TCP/Fack, TCP/Newreno, TCP/Reno, TCP/Sack1, or
TCP/FullTcp
traffic_type : either exp, pareto, cbr, FTP, or Telnet
destination_type : either TCPSink, TCPSink/DelAck, TCPSink/Sack1, or TCPSink/Sack1/DelAck

simulation _time : is the same time that used in the scenario file and in the simulation environment.

Note

The above instruction is sensitive for the case of the letter (the agent type, traffic type and destination type must be written as found above).

Simulation Environment

In order to improve the activity of the modified file, the simulation had performed depending on the NS-2 and the mobile ad-hoc network environment that shown in table1.

Simulation Results

Figure1 represents the number of the dropped packets for each type of the TCP, figure 2 illustrates the throughput of mobile ad-hoc network with each type of the TCP, figure 3 shows the average jitter for every type of the TCP, and the normalize routing load shown in figure 4.

Table 1: Mobile ad-hoc network environment

The parameter	The value
Operating system	Debian Linux
Network simulator	NS-2.34
Number of nodes	10 nodes
Simulation time	100 second
Routing protocol	DSDV
Simulation area	700m * 700m
Pause time	5s
Pause time distribution	Uniform
Speed	6m/s
Speed distribution	Uniform
Agent type (source type)	TCP, TCP/Vegas, TCP/Linux, TCP/Fack, TCP/Newreno, TCP/Reno, TCP/Sack1, and FullTcp
TCP sink type	TCPSink
Traffic type	FTP
Movement model	Random way point model
Sending rate	3 packets/second
Size of packet	512 bytes/packets
Max connection	4

CONCLUSION

The old traffic generator file was restricted for little types of traffic that studied in many researches before this paper. The modified file can generate many new types of traffic between source and destination nodes efficiently to be used in the simulation of the network that depend on the NS-2. This code can be used by any researcher that depends on the NS-2 to study the behaviour of these types of TCP as sender with many types as receiver.

REFERENCES

1. Jyotsna Rathee and A. K. Verma, "Simulation, Analysis and Comparison of DSDV Protocol in MANETS", MSc. thesis, Computer Science and Engineering Department, Thapar University (2009).
2. FJ Arbona Bernat and P.F.A. Van Mieghem, "Simulation of Ant Routing Protocol for Ad-hoc networks in NS-2", MSc. thesis, Delft university of technology, Faculty of Electrical Engineering, Mathematics and Computer Science Network Architectures and Services Group (2006).
3. Teerawat Issariyakul and Ekram Hossain, "Introduction to Network Simulator NS2", springer, ISBN: 978-0-387-71759-3 (2009).
4. P.Periyasamy and E.Karthikeyan, "Performance evaluation of aomdv protocol based on various scenario and traffic patterns", *International Journal of Computer Science, Engineering and Applications* (IJCSEA) **1**6): (2011).
5. YAMSANI RAVIKUMAR and SARATH KUMAR CHITTAMURU, "A Case Study on MANET Routing Protocols Performance over TCP and HTTP", MSc. thesis, School of Engineering blekinge Institute of Technology, Sweden (2010).
6. Kevin Fall and Kannan Varadhan, "The ns

Manual (formerly ns Notes and Documentation)", 2010.

7. Saad Talib Hasson, Ahmed Jawad Kadhim, Zainab Saad Talib, "Enhancing the NS-2 Traffic Generator for the MANETs", IOSR Journal of Computer Engineering (IOSRJCE), ISSN: 2278-0661 Volume 4, Issue 2, PP 12-16, 2012.