

NS-2 Simulation Report

Vasudha Devarakonda

Tushar Premanand

Test Cases:

Case 1:

src1-R1 and R2-rcv1 end-2-end delay = 5 ms

src2-R1 and R2-rcv2 end-2-end delay = 12.5 ms

Case 2:

src1-R1 and R2-rcv1 end-2-end delay = 5 ms

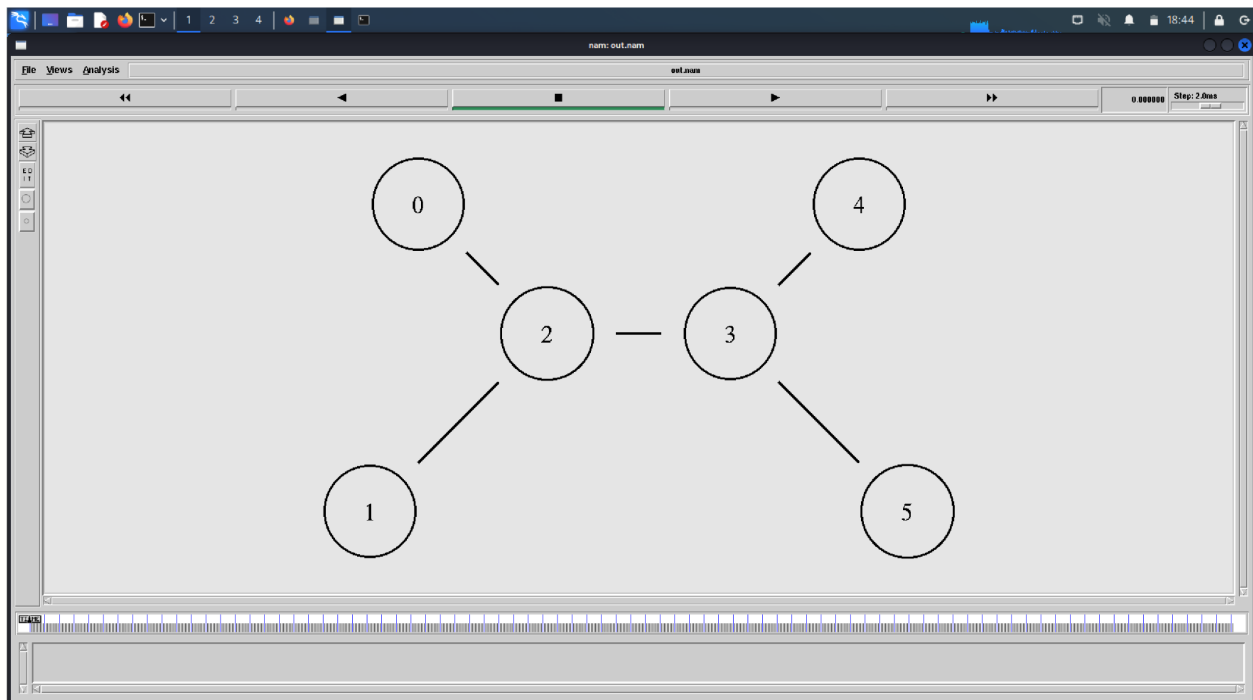
src2-R1 and R2-rcv2 end-2-end delay = 20 ms

Case 3:

src1-R1 and R2-rcv1 end-2-end delay = 5 ms

src2-R1 and R2-rcv2 end-2-end delay = 27.5 ms

Layout :



Code:

```
if { $argc != 2 } {  
    puts "Invalid number of arguments/ Incorrect format"  
    puts "Enter ns $argv0 <TCP_Flavour> <case_num>"  
    exit  
}
```

```

set flavor [lindex $argv 0]
set case_no [lindex $argv 1]

if { $case_no > 3 || $case_no < 1 } {
    puts "Invalid case num entered $case"
    puts "Enter the case as 1 for src1-R1 and R2-rcv1 end-2-end
delay = 5 ms and src2-R1 and R2-rcv2 delay = 12.5 ms"
    puts "Enter the case as 2 for src1-R1 and R2-rcv1 end-2-end
delay = 5 ms and src2-R1 and R2-rcv2 delay = 20 ms"
    puts "Enter the case as 3 for src1-R1 and R2-rcv1 end-2-end
delay = 5 ms and src2-R1 and R2-rcv2 delay = 27.5 ms"
    exit
}

global delay
global flavr
set delay 0

switch $case_no {
    global delay
    1 {set delay "12.5ms"}
    2 {set delay "20ms"}
    3 {set delay "27.5ms"}
}

if {$flavor == "SACK"} {
    set flavr "Sack1"
} elseif {$flavor == "VEGAS"} {
    set flavr "Vegas"
} else {
    puts "Invalid TCP Flavor $flavor"
    exit
}

#create ns simulator
set ns [new Simulator]
set file "out_${flavor}$case_no"

set f1 [open out1.tr w]
set f2 [open out2.tr w]

#Opening ns tracefile

```

```
set tracefile [open out.tr w]
$ns trace-all $tracefile

#Open NAM trace file
set namfile [open out.nam w]
$ns namtrace-all $namfile

#create the 6 nodes
set src1 [$ns node]
set src2 [$ns node]
set R1 [$ns node]
set R2 [$ns node]
set rcv1 [$ns node]
set rcv2 [$ns node]

#Define colours for data flows
$ns color 1 Red
$ns color 2 Green

set thrput1 0
set thrput2 0
set counter 0

# Setup a TCP connection
set tcp1 [new Agent/TCP/$flavr]
set tcp2 [new Agent/TCP/$flavr]
$ns attach-agent $src1 $tcp1
$ns attach-agent $src2 $tcp2

$tcp1 set class_ 1
$tcp2 set class_ 2

# Link Definition
$ns duplex-link $src1 $R1 10.0Mb 5ms DropTail
$ns duplex-link $rcv1 $R2 10.0Mb 5ms DropTail
$ns duplex-link $R1 $R2 1.0Mb 5ms DropTail

$ns duplex-link $src2 $R1 10.0Mb $delay DropTail
$ns duplex-link $rcv2 $R2 10.0Mb $delay DropTail

#node positions && flow of link
$ns duplex-link-op $R1 $R2 orient right
$ns duplex-link-op $src1 $R1 orient right-down
$ns duplex-link-op $src2 $R1 orient right-up
```

```

$ns duplex-link-op $R2 $rcv1 orient right-up
$ns duplex-link-op $R2 $rcv2 orient right-down

#Applications Definition
#set a FTP application over TCP connection
set ftp1 [new Application/FTP]
set ftp2 [new Application/FTP]

$ftp1 attach-agent $tcp1
$ftp2 attach-agent $tcp2

set sink1 [new Agent/TCPSink]
set sink2 [new Agent/TCPSink]
$ns attach-agent $rcv1 $sink1
$ns attach-agent $rcv2 $sink2

$ns connect $tcp1 $sink1
$ns connect $tcp2 $sink2

#Finish procedure
proc finish {} {
    global ns nf tracefile namfile file thrput1 thrput2 counter
    # global f1
    $ns flush-trace
    #close $f1
    puts "Avg throughput for Src1=[expr $thrput1/$counter]
Mbits/sec\n"
    puts "Avg throughput for Src2=[expr $thrput2/$counter]
Mbits/sec\n"
    close $tracefile
    close $namfile
    exec nam out.nam &
    #exec xgraph out1.tr out2.tr-geometry 800x400 &
    exit 0
}

proc record {} {
    global sink1 sink2 f1 f2 thrput1 thrput2 counter
    #Get an instance of the simulator
    set ns [Simulator instance]
    #Set the time after which the procedure should be called again
    set time 0.5

```

```

    #bytes received by the
    set bw1 [$sink1 set bytes_]
set bw2 [$sink2 set bytes_]

    #Get the current time
    set now [$ns now]

    #bandwidth calculation
    puts $f1 "$now [expr $bw1/$time*8/1000000] "
    puts $f2 "$now [expr $bw2/$time*8/1000000] "
    set thrput1 [expr $thrput1+ $bw1/$time*8/1000000]
    set thrput2 [expr $thrput2+ $bw2/$time*8/1000000]
    set counter [expr $counter + 1]

    #Rese the bytes on the traffic sinks
    $sink1 set bytes_ 0
    $sink2 set bytes_ 0

    $ns at [expr $now+$time] "record"
}

$ns at 0 "record"
$ns at 0 "$ftp1 start"
$ns at 0 "$ftp2 start"
$ns at 400 "$ftp1 stop"
$ns at 400 "$ftp2 stop"
$ns at 400 "finish"
$ns color 1 Blue
$ns color 2 Red
set tf1 [open "$file--S1.tr" w]
$ns trace-queue $src1 $R1 $tf1

set tf2 [open "$file--S2.tr" w]
$ns trace-queue $src2 $R1 $tf2

$ns run

```

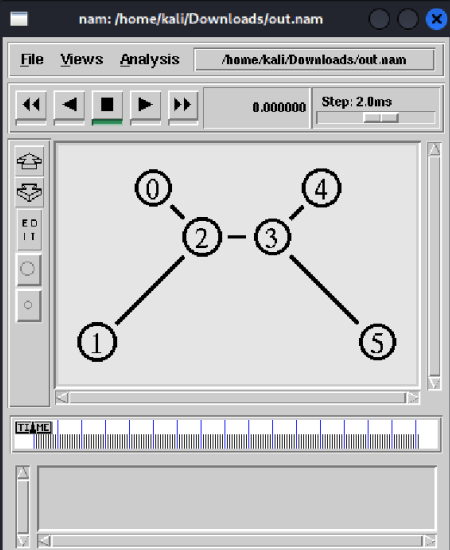
1. For flavor – SACK cases 1,2,3 :

```
(kali@kali)~/Downloads
$ ns ns2.tcl SACK 1
Avg throughput for Src1=0.5233080000000588 MBits/sec
Avg throughput for Src2=0.47523919999999636 MBits/sec

(kali@kali)~/Downloads
$ ns ns2.tcl SACK 2
Avg throughput for Src1=0.54514800000000518 MBits/sec
Avg throughput for Src2=0.45339919999999467 MBits/sec

(kali@kali)~/Downloads
$ ns ns2.tcl SACK 3
Avg throughput for Src1=0.5650328000000007 MBits/sec
Avg throughput for Src2=0.43349359999999548 MBits/sec

(kali@kali)~/Downloads
$
```



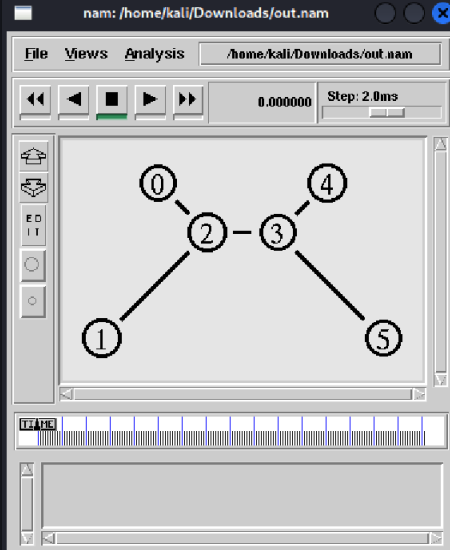
2. For flavor – VEGAS cases 1,2,3 :

```
(kali@kali)~/Downloads
$ ns ns2.tcl VEGAS 1
Avg throughput for Src1=0.58252000000000148 MBits/sec
Avg throughput for Src2=0.41599999999999765 MBits/sec

(kali@kali)~/Downloads
$ ns ns2.tcl VEGAS 2
Avg throughput for Src1=0.68663999999999692 MBits/sec
Avg throughput for Src2=0.31185999999999814 MBits/sec

(kali@kali)~/Downloads
$ ns ns2.tcl VEGAS 3
Avg throughput for Src1=0.74901999999999647 MBits/sec
Avg throughput for Src2=0.24948000000000081 MBits/sec

(kali@kali)~/Downloads
$
```



TCP throughput for each case :

VEGAS

	Src1 throughput	Src2 throughput	Ratio -> src1/src2
Case 1	0.582	0.415	1.402
Case 2	0.6866	0.311	2.207
Case 3	0.749	0.249	3.008

SACK

	Src1 throughput	Src2 throughput	Ratio -> src1/src2
Case 1	0.5233	0.475	1.101
Case 2	0.545	0.4533	1.2022
Case 3	0.565	0.4334	1.303

Why does one of the TCP flavors perform better than the other for Case 1 ?

TCP VEGAS is more stable than SACK.

SACK uses packet loss to denote congestion, so the sender continuously increases the sending rate until there is congestion and then they cut back. This cycle continues causing the system to keep oscillating.

TCP VEGAS flattens out the sending rate at the optimal bandwidth utilization point thus inducing stability in the system. TCP VEGAS has a good estimation of incipient congestion and efficient estimation of congestion by measuring change in throughput rather than packet loss.