



**ELECTRICAL & COMPUTER  
ENGINEERING**  
T E X A S   A & M   U N I V E R S I T Y

---

**SHASHANK KAMATH KALASA MOHANDAS**

UIN: 627003580

**SRINIVAS PRAHALAD SUMUKHA**

UIN: 627008254

---

**TEAM 6**  
**ASSIGNMENT 5**

Computer Communication and Networks  
ECEN 602  
FALL 2018

## Role of SHASHANK KAMATH KALASA MOHANDAS

- ns2.tcl
- README file
- run.sh

## Role of SRINIVAS PRAHALAD SUMUKHA

- ns2.tcl
- README file
- run.sh

### ***How to run:***

1. Open the run.sh file and set the alias xgraph=/path/to/xgraph/folder/bin/xgraph
2. Run the shell script in the terminal by typing the command './run.sh'

### ***Files included in this assignment:***

- ns2.tcl
- README
- run.sh

### **run.sh:**

```
#####  
  
#Team 6 NS2 simulator  
  
#####  
  
ns ns2_1.tcl VEGAS 1  
// ns ns2_1.tcl VEGAS 2  
// ns ns2_1.tcl VEGAS 3  
// ns ns2_1.tcl SACK 1  
// ns ns2_1.tcl SACK 2  
// ns ns2_1.tcl SACK 3  
  
alias xgraph=/home/sumukh110/ns2/XGraph4.38_linux64/bin/xgraph  
  
xgraph out1.tr out2.tr
```

## ns2.tcl Implementation:

### *ns2.tcl*

```
global version, case, delay, ver
set delay =0
set version [lindex $argv 0]
set case [lindex $argv 1]

if {$case == 1} {
    set delay "12.5ms"
} elseif {$case == 2} {
    set delay "20ms"
} elseif {$case == 3} {
    set delay "27.5ms"
} else {
    puts "Invalid Case"
    exit
}

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

set ns [new Simulator]

set out1 [open out1.tr w]
set out2 [open out2.tr w]
set out3 [open out3.tr w]
#set file "out_$flavor$case"
set nf [open out.tr w]
$ns trace-all $nf

set namf [open out.nam w]
$ns namtrace-all $namf

#NODE INTITIALIZING
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 different colors for data flows
```

```
$ns color 1 Blue
```

```
$ns color 2 Red
```

```
set tcp1 [new Agent/TCP/$ver]
```

```
set tcp2 [new Agent/TCP/$ver]
```

```
$ns attach-agent $src1 $tcp1
```

```
$ns attach-agent $src2 $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
```

```
#Createlinks between nodes
```

```
$ns duplex-link $r1 $r2 1.0Mb 5ms DropTail
```

```
$ns duplex-link $src1 $r1 10.0Mb 5ms DropTail
```

```
$ns duplex-link $rcv1 $r2 10.0Mb 5ms DropTail
```

```
$ns duplex-link $src2 $r1 10.0Mb $delay DropTail
```

```
$ns duplex-link $rcv2 $r2 10.0Mb $delay DropTail
```

```
#Give node position (for NAM)
```

```
$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
```

```
#Creating FTP
```

```
set ftp1 [new Application/FTP]
```

```
set ftp2 [new Application/FTP]
```

```
$ftp1 attach-agent $tcp1
```

```
$ftp2 attach-agent $tcp2
```

```
#Initializing
```

```
set sum1 0
```

```
set sum2 0
```

```
set count 0
```

```
#Calculating
```

```
proc calc {} {
```

```
    global ns sink1 sink2 out1 out2 out3 sum1 sum2 count
```

```
    set bandwidth1 [$sink1 set bytes_]
```

```
    set bandwidth2 [$sink2 set bytes_]
```

```

set time 0.5

set now [$ns now]

#Initialization
if {$now == 100} {
    $sink1 set bytes_ 0
    $sink2 set bytes_ 0
}
#Throughput between 100 and 400 seconds
if {$now > 100 && $now <= 400} {
    set throughput1 [expr $bandwidth1/$time *8/1000000]
    set throughput2 [expr $bandwidth2/$time *8/1000000]
    set sum1 [expr $sum1 + $throughput1]
    set sum2 [expr $sum2 + $throughput2]
    set count [expr $count + 1]
    set ratio [expr $throughput1/$throughput2]

    #Write values to output files
    puts $out1 "$now $throughput1"
    puts $out2 "$now $throughput2"
    puts $out3 "$ratio"
    $sink1 set bytes_ 0
    $sink2 set bytes_ 0
}

if { $now == 400.5 } {
    set averagethroughput1 [ expr $sum1/$count]
    set averagethroughput2 [ expr $sum2/$count]
    puts "Average throughput for src1 : $averagethroughput1 MBits/sec"
    puts "Average throughput for src2 : $averagethroughput2 MBits/sec"
    set ratio [expr $averagethroughput1/$averagethroughput2]
    puts "Ratio of throughputs : $ratio"
}
#Recursion call
$ns at [expr $now + $time] "calc"
}

#Terminating
proc finish {} {
    global ns nf namf
    $ns flush-trace
    close $nf
    close $namf
    #exec nam out.nam &
    exec xgraph out1.tr out2.tr
    #-geometry 800x400 &
    exit 0
}

```

```
}  
  
#Activity  
$ns at 0.5 "$ftp1 start"  
$ns at 0.5 "$ftp2 start"  
  
#Call calc  
$ns at 100 "calc"  
$ns at 401 "$ftp1 stop"  
$ns at 401 "$ftp2 stop"  
  
#Call Finish  
$ns at 405 "finish"  
$ns run
```

## **TEST CASES:**

**There are 3 test cases which are presented in this section for VEGAS:**

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 = 12.5 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

Similarly, there are 3 test cases which are presented in this section for SACK:

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:

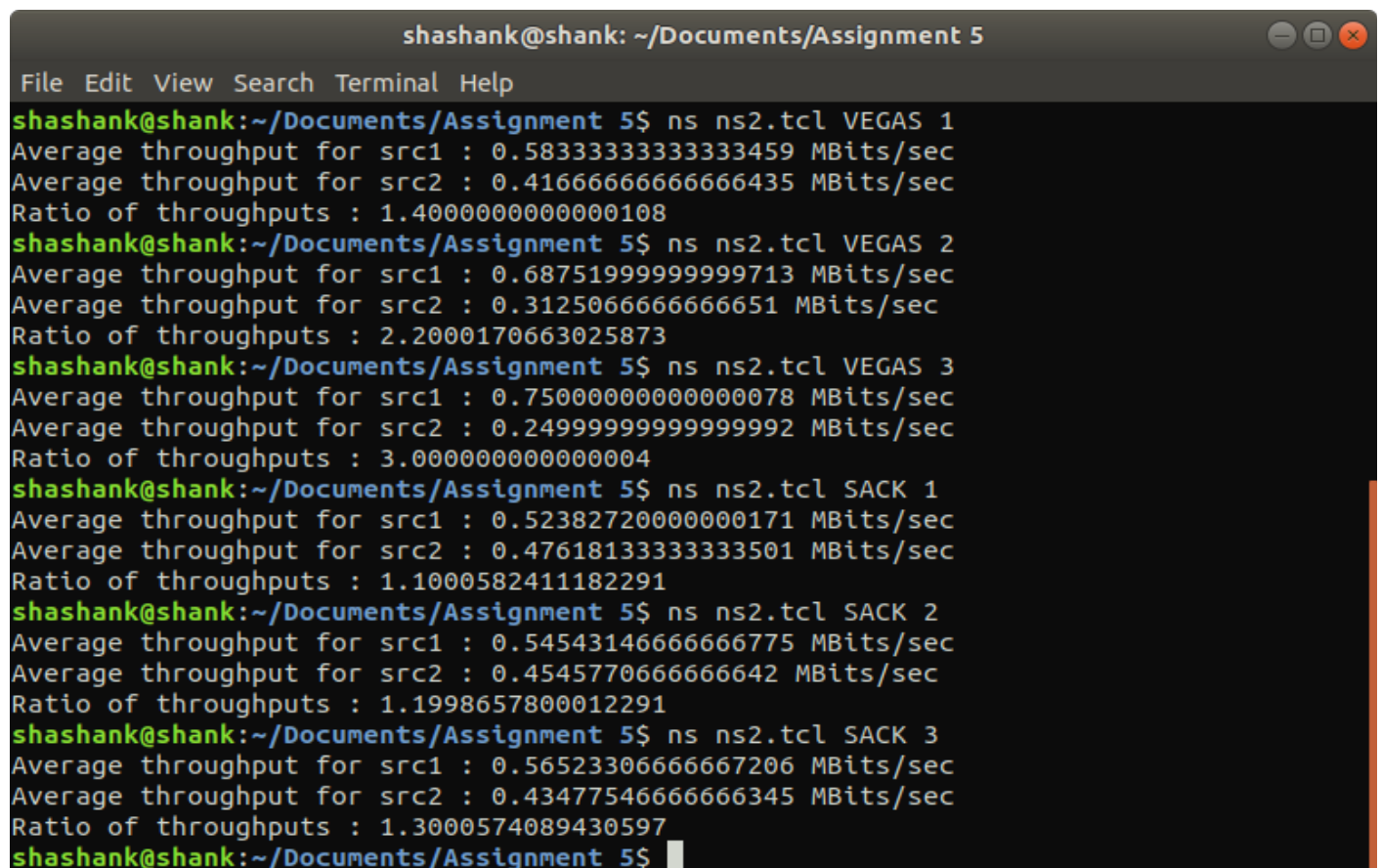
src1-R1 and R2-rcv1 end-2-end delay = 5 ms

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

Case 3:

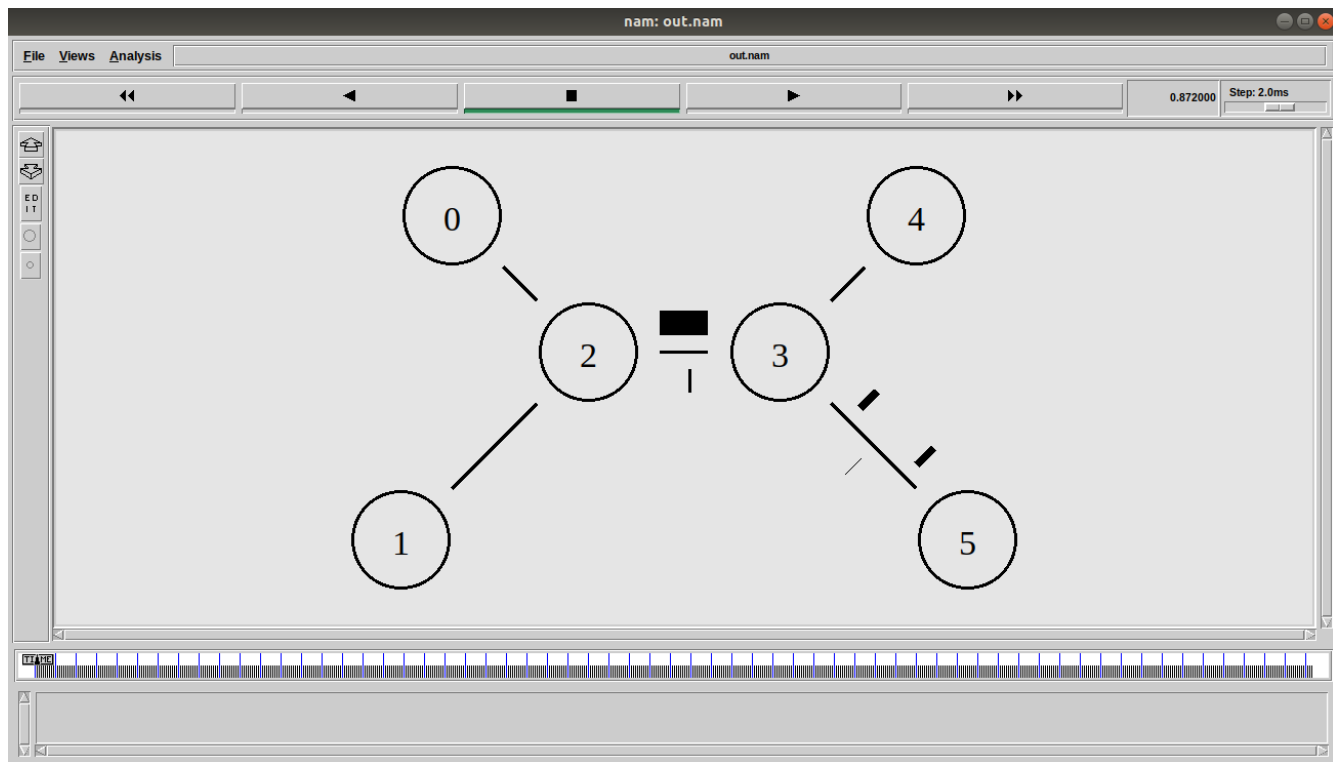
src1-R1 and R2-rcv1 end-2-end delay = 5 ms

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



```
shashank@shank: ~/Documents/Assignment 5
File Edit View Search Terminal Help
shashank@shank:~/Documents/Assignment 5$ ns ns2.tcl VEGAS 1
Average throughput for src1 : 0.58333333333333459 MBits/sec
Average throughput for src2 : 0.416666666666666435 MBits/sec
Ratio of throughputs : 1.40000000000000108
shashank@shank:~/Documents/Assignment 5$ ns ns2.tcl VEGAS 2
Average throughput for src1 : 0.68751999999999713 MBits/sec
Average throughput for src2 : 0.31250666666666651 MBits/sec
Ratio of throughputs : 2.2000170663025873
shashank@shank:~/Documents/Assignment 5$ ns ns2.tcl VEGAS 3
Average throughput for src1 : 0.75000000000000078 MBits/sec
Average throughput for src2 : 0.24999999999999992 MBits/sec
Ratio of throughputs : 3.0000000000000004
shashank@shank:~/Documents/Assignment 5$ ns ns2.tcl SACK 1
Average throughput for src1 : 0.523827200000000171 MBits/sec
Average throughput for src2 : 0.476181333333333501 MBits/sec
Ratio of throughputs : 1.1000582411182291
shashank@shank:~/Documents/Assignment 5$ ns ns2.tcl SACK 2
Average throughput for src1 : 0.545431466666666775 MBits/sec
Average throughput for src2 : 0.45457706666666642 MBits/sec
Ratio of throughputs : 1.1998657800012291
shashank@shank:~/Documents/Assignment 5$ ns ns2.tcl SACK 3
Average throughput for src1 : 0.565233066666667206 MBits/sec
Average throughput for src2 : 0.434775466666666345 MBits/sec
Ratio of throughputs : 1.3000574089430597
shashank@shank:~/Documents/Assignment 5$
```

Screenshot shows the result of all the above-mentioned cases.



## RESULT TABULATION:

Flavour	Case number	Throughput src1 Mbps	Throughput src2 Mbps	Ratio
VEGAS	1	0.5833	0.4166	1.4
VEGAS	2	0.6875	0.3125	2.2
VEGAS	3	0.750	0.249	3
SACK	4	0.5238	0.4761	1.1
SACK	5	0.5454	0.4545	1.19
SACK	6	0.5652	0.43477	1.3

From the table we can infer that SACK has almost no impact of RTT. The RTT for the above-mentioned cases were of the ratio 1:2, 1:3 and 1:4, but the ratio of throughput is 1.1, 1.19 and 1.3, which do not match with the RTT rates. On the other hand, VEGAS's throughput ratios are 1.4, 2.2 and 3, on comparing the ratio of throughput with the RTT for the above-mentioned three cases, we can see that the RTT and throughput ratio match.



Hence, we can conclude that VEGAS performs better, or the throughput of VEGAS increases with decrease in RTT.

If we make a comparison between the ratio of throughputs of VEGAS and SACK, we can see that VEGAS constantly outperforms SACK in all the three cases. There are several more reasons for VEGAS's better performance when compared to SACK other than the one mentioned above.

Firstly, VEGAS has a good way of efficient estimation of congestion, i.e. VEGAS relies on change in throughput rather than packet loss.

Secondly, SACK uses packet loss as a parameter to gauge the congestion or recognize the congestion in the network, So the sender keeps increasing the rate of transmission until congestion. After the packet loss the SACK realises the congestions and reduces the rate. This makes the system to oscillate making SACK unstable when compared to VEGAS.

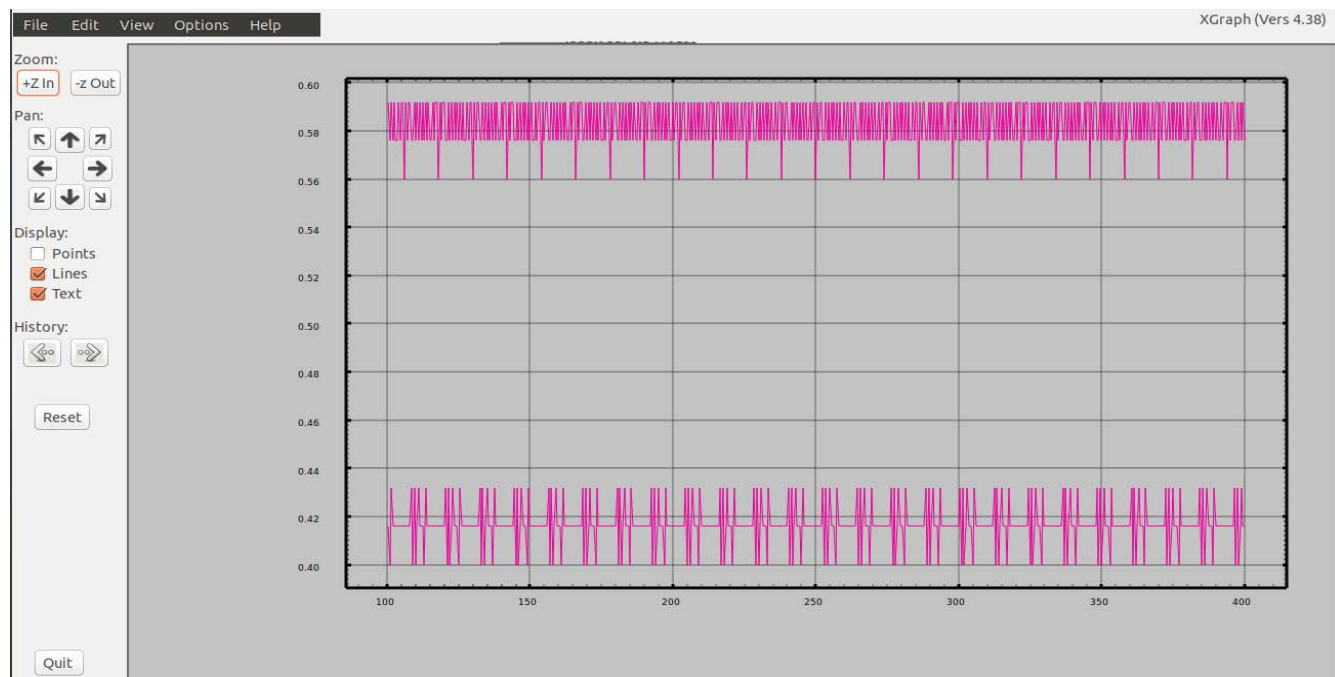
Thirdly, SACK as mentioned in the TB is harder to incorporate in the current TCP.

## TEST CASES:

### 1. VEGAS 1: Case 1

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

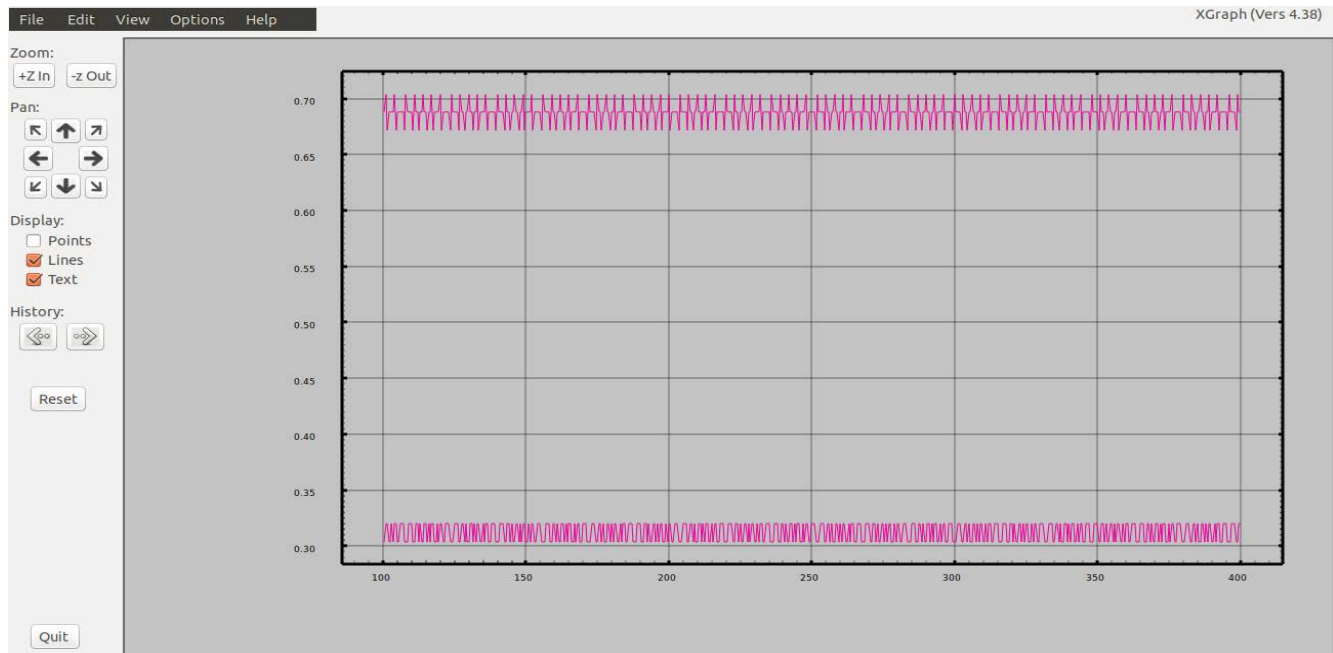
src2-R1 and R2-rcv2 end-2-end delay = 12.5 ms



## 2. VEGAS 2: Case 2

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

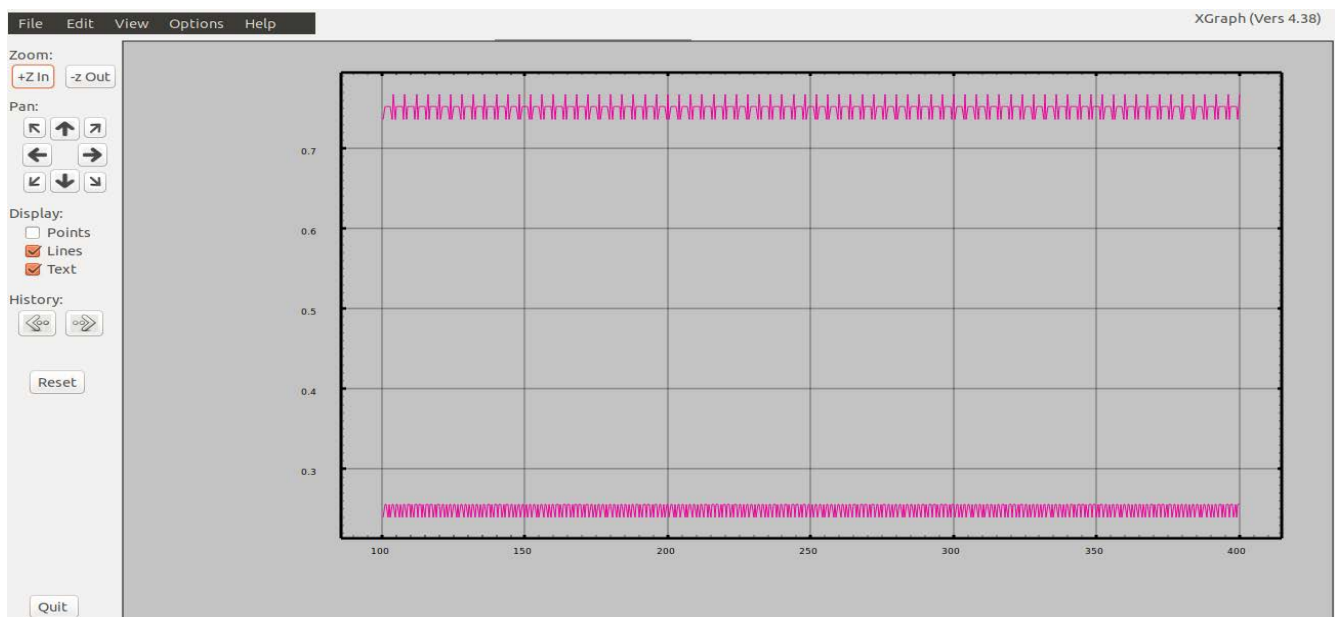
src2-R1 and R2-rcv2 end-2-end delay = 20 ms



## 3. VEGAS 3: Case 3

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

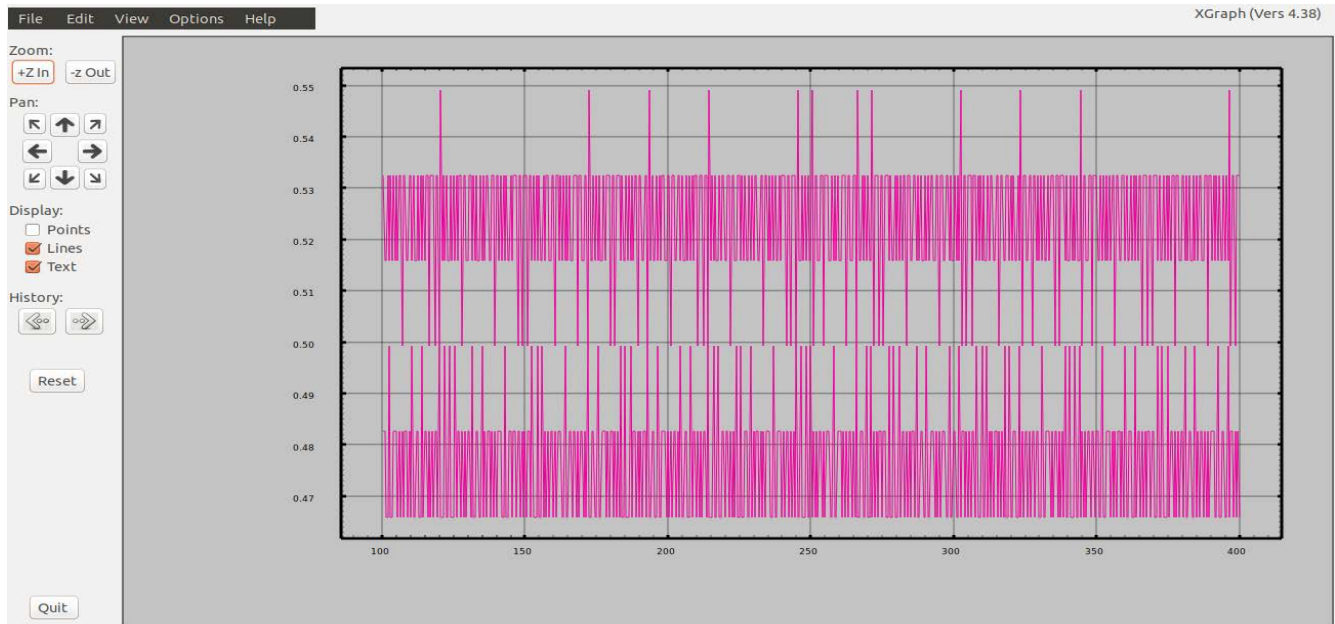
src2-R1 and R2-rcv2 end-2-end delay = 27.5 ms



#### 4. SACK 1: Case 4

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

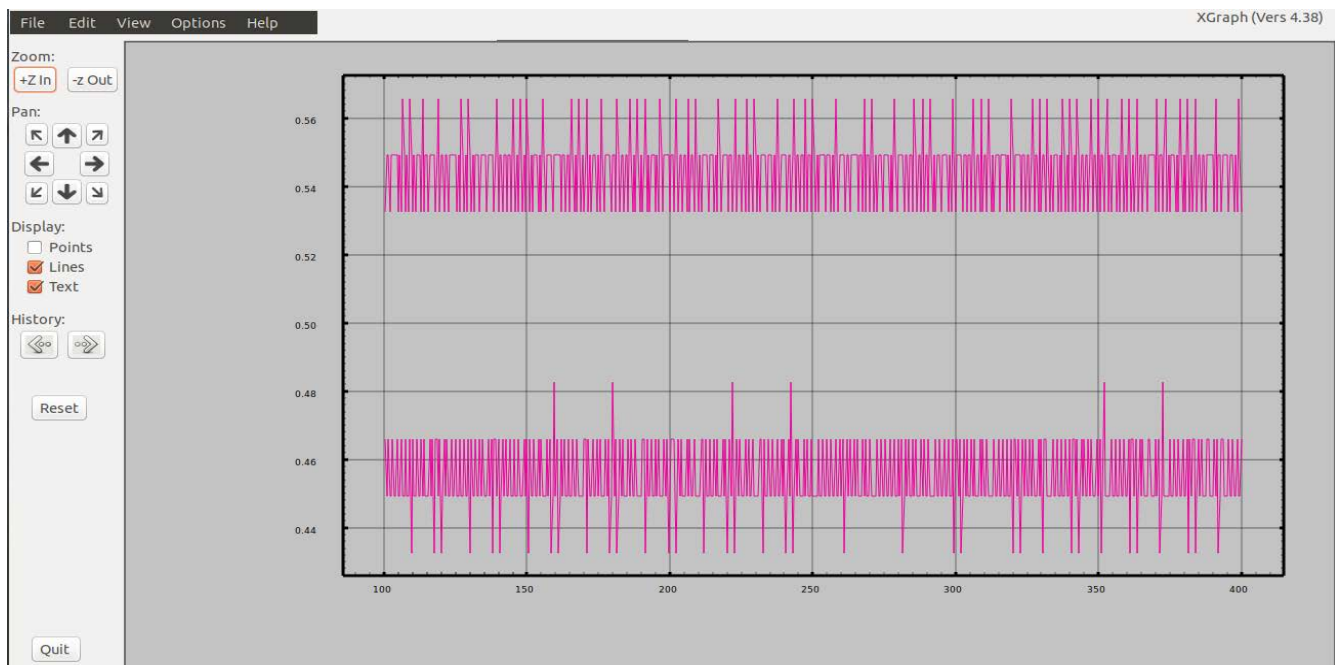
src2-R1 and R2-rcv2 end-2-end delay = 12.5 ms



#### 5. SACK 2: Case 5

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

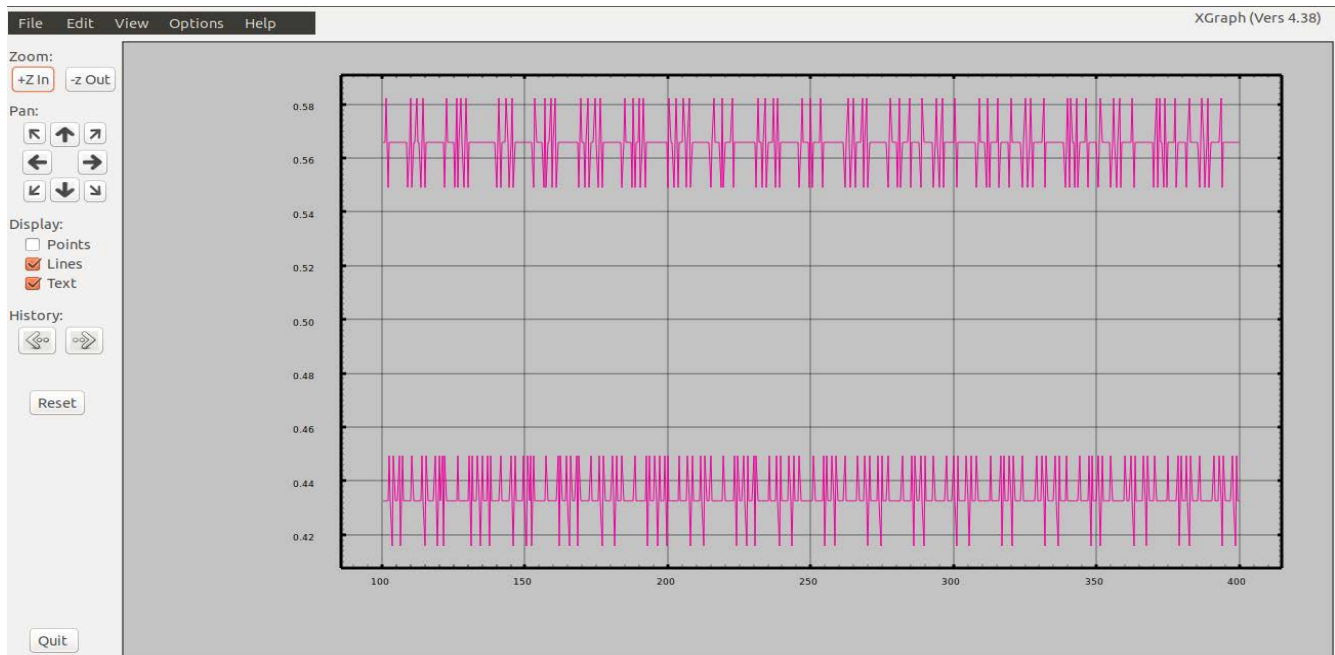
src2-R1 and R2-rcv2 end-2-end delay = 20 ms



## 6. SACK 3: Case 6

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

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



Reference:

<http://inst.eecs.berkeley.edu/~ee122/fa05/projects/Project2/SACKRENEVEGAS.pdf>