NS 2 . Assignment 5 – ECEN 602
Group 20
Himanshu Gupta
Adarsh

1. TCP VEGAS

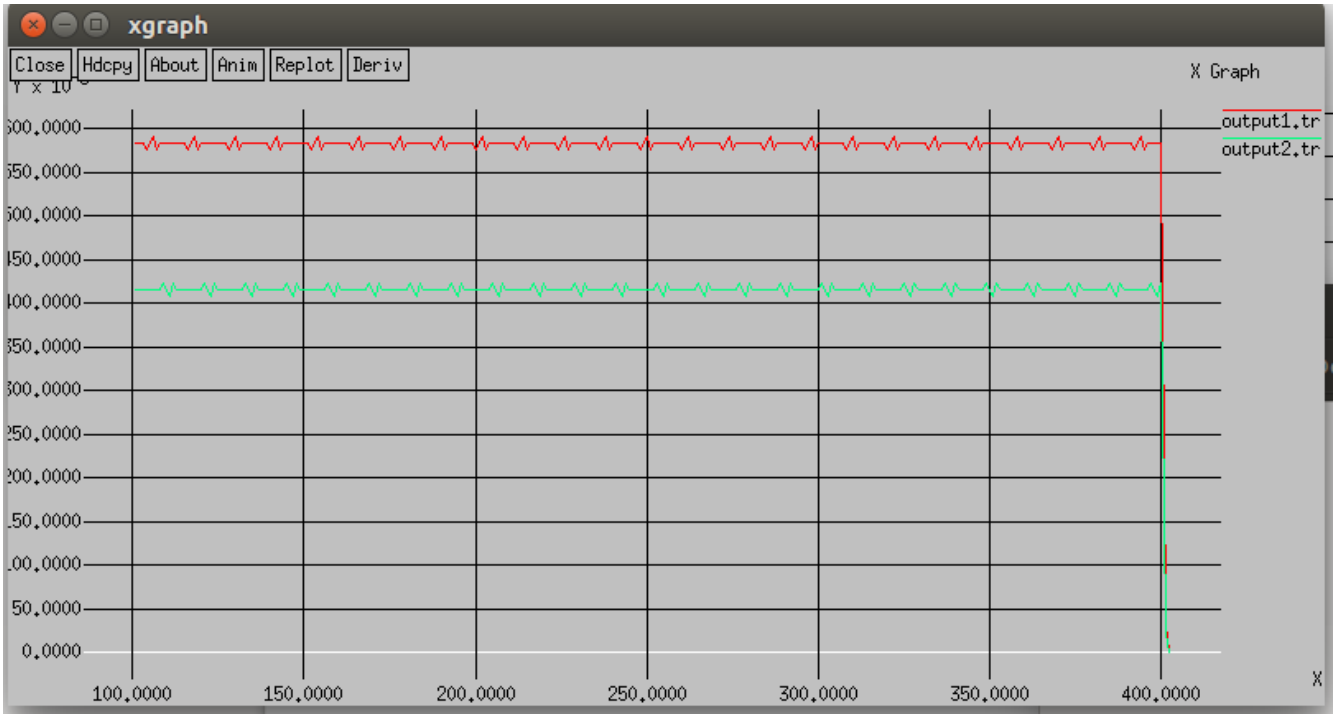
Test Case	Throughput ratio src1/src2
1	1.403
2	2.200
3	3.00

a) case 1

FLOW	THROUGHPUT (Mbps)
(src 1 – rcv 1)	0.5833
(src 2 – rcv 2)	0.4166

Throughput ratio – 1.403

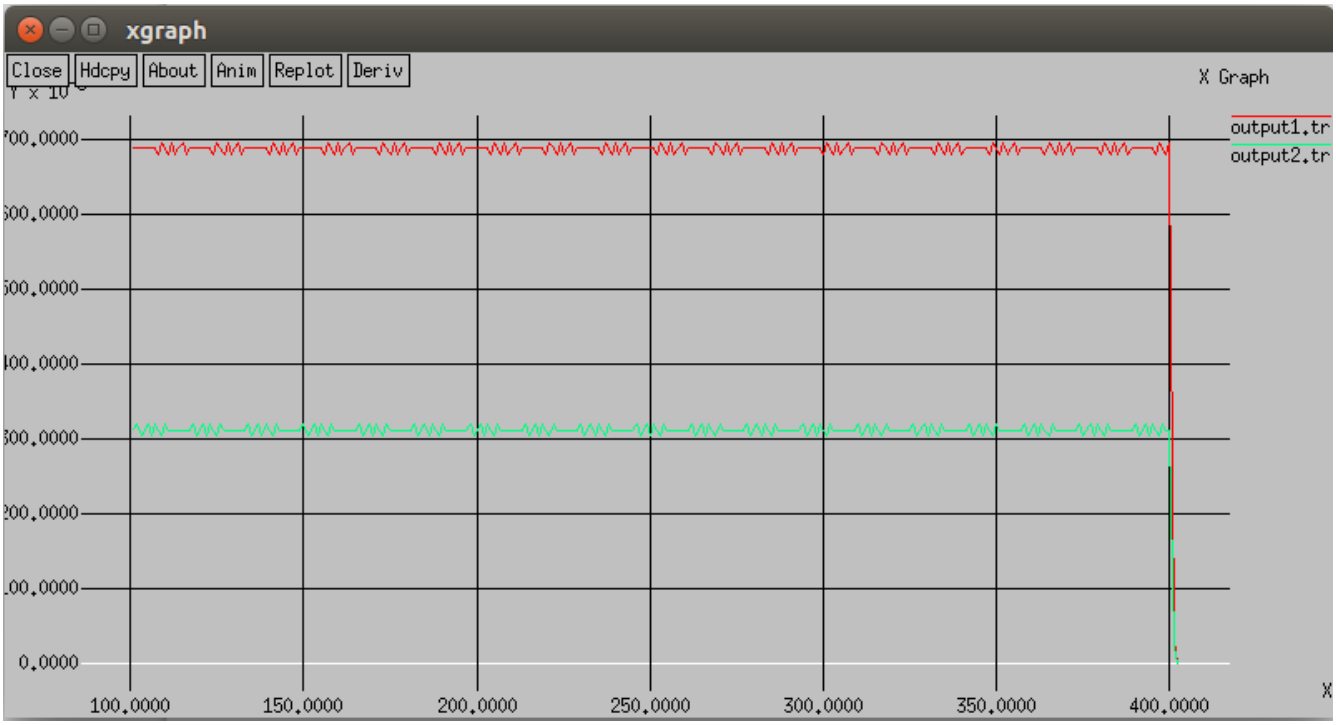
xGraph



b) case 2

FLOW	THROUGHPUT(Mbps)
(src 1 – rcv 1)	0.6875
(src 2 – rcv 2)	0.3125

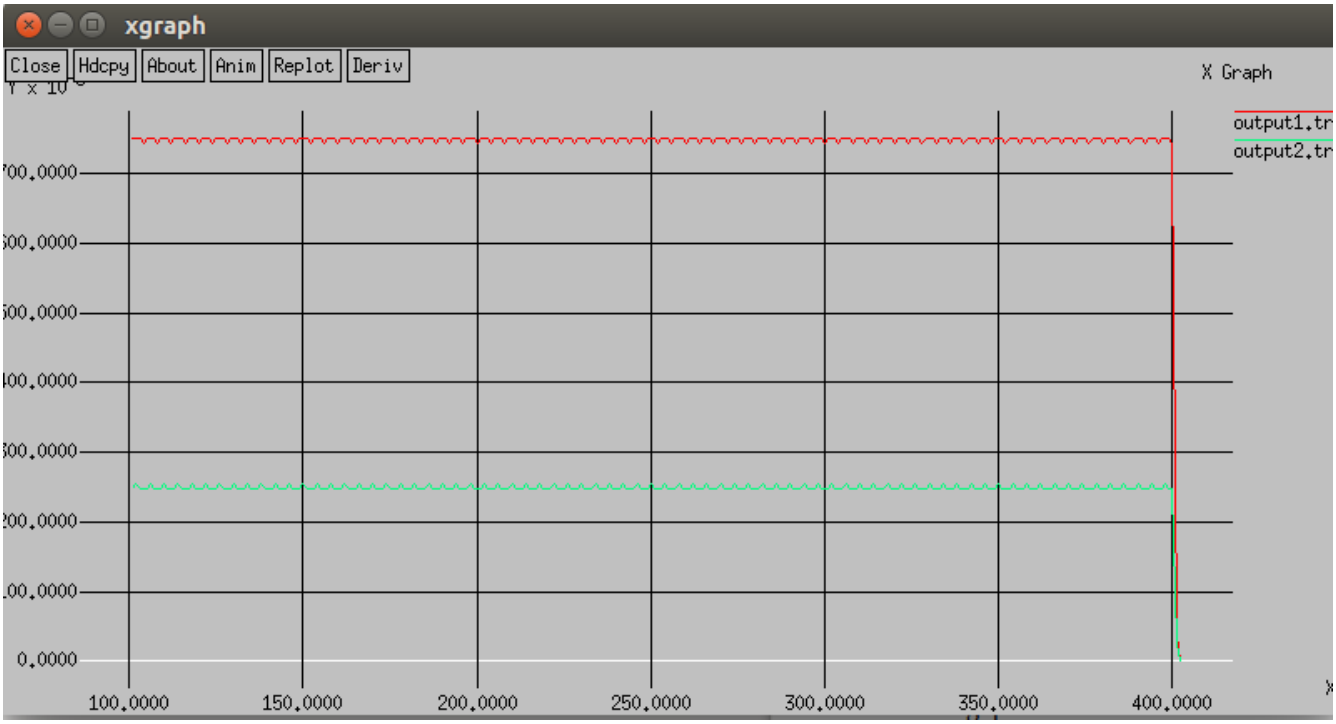
Throughput ratio – 2.200



c) case 3

FLOW	THROUGHPUT(Mbps)
(src 1 – rcv 1)	0.7500
(src 2 – rcv 2)	0.2499

Throughput ratio – 3.00



2. SACK

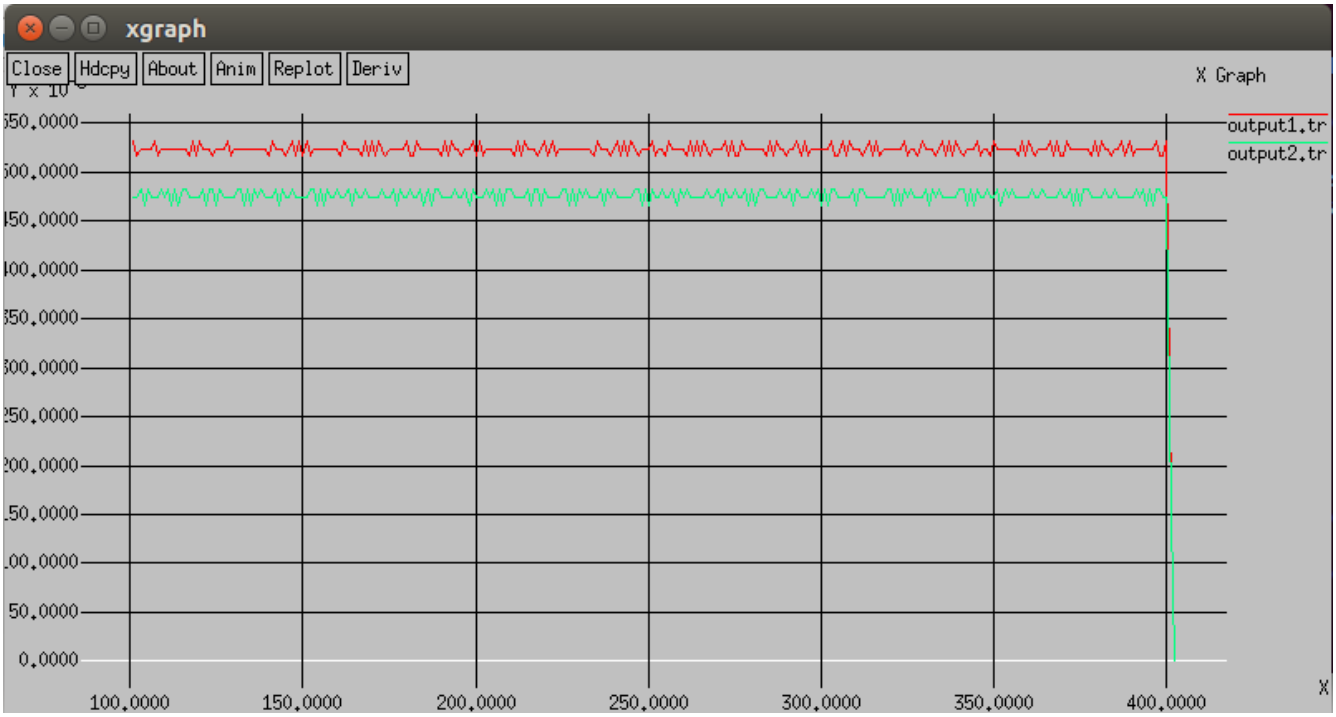
Case	Throughput ratio src1/src2
1	1.1
2	1.19
3	1.3

a) case 1

FLOW	THROUGHPUT (Mbps)

(src 1 – rcv 1)	0.5238
(src 2 – rcv 2)	0.4761

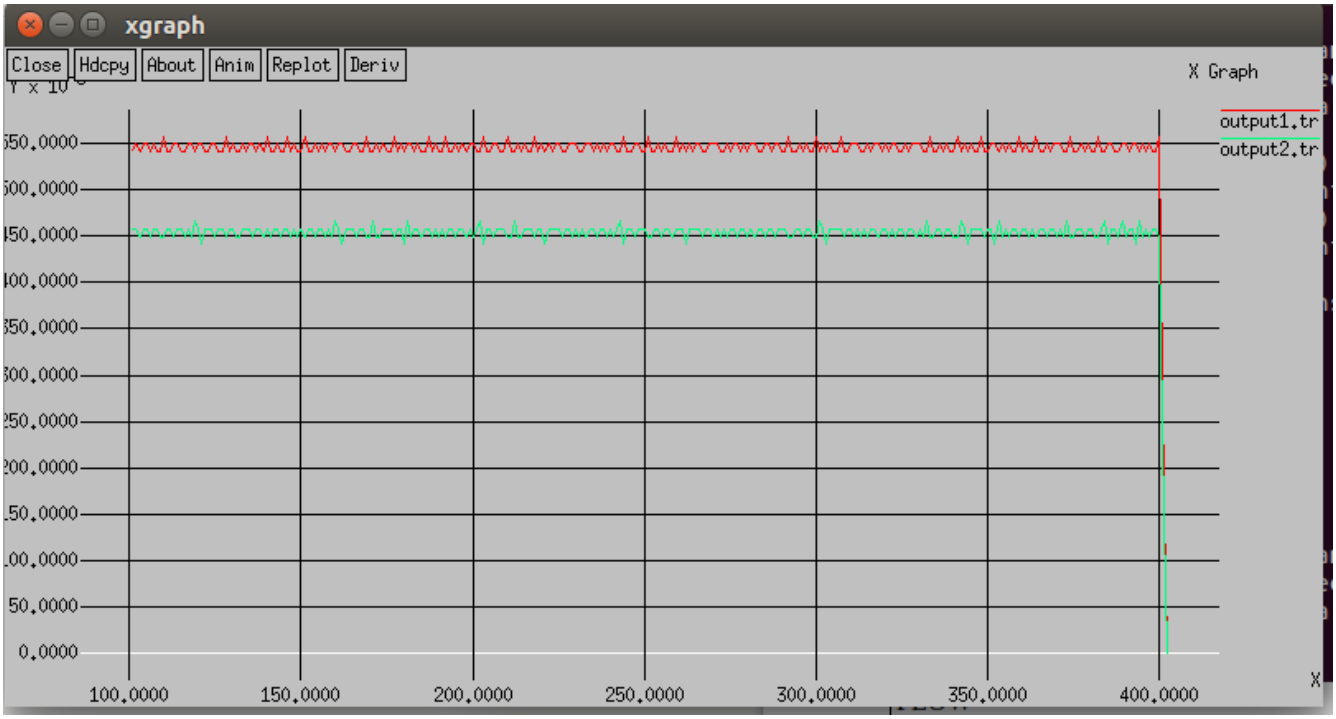
Throughput ratio – 1.100



b) case 2

FLOW	THROUGHPUT(Mbps)
(src 1 – rcv 1)	0.5454
(src 2 – rcv 2)	0.4545

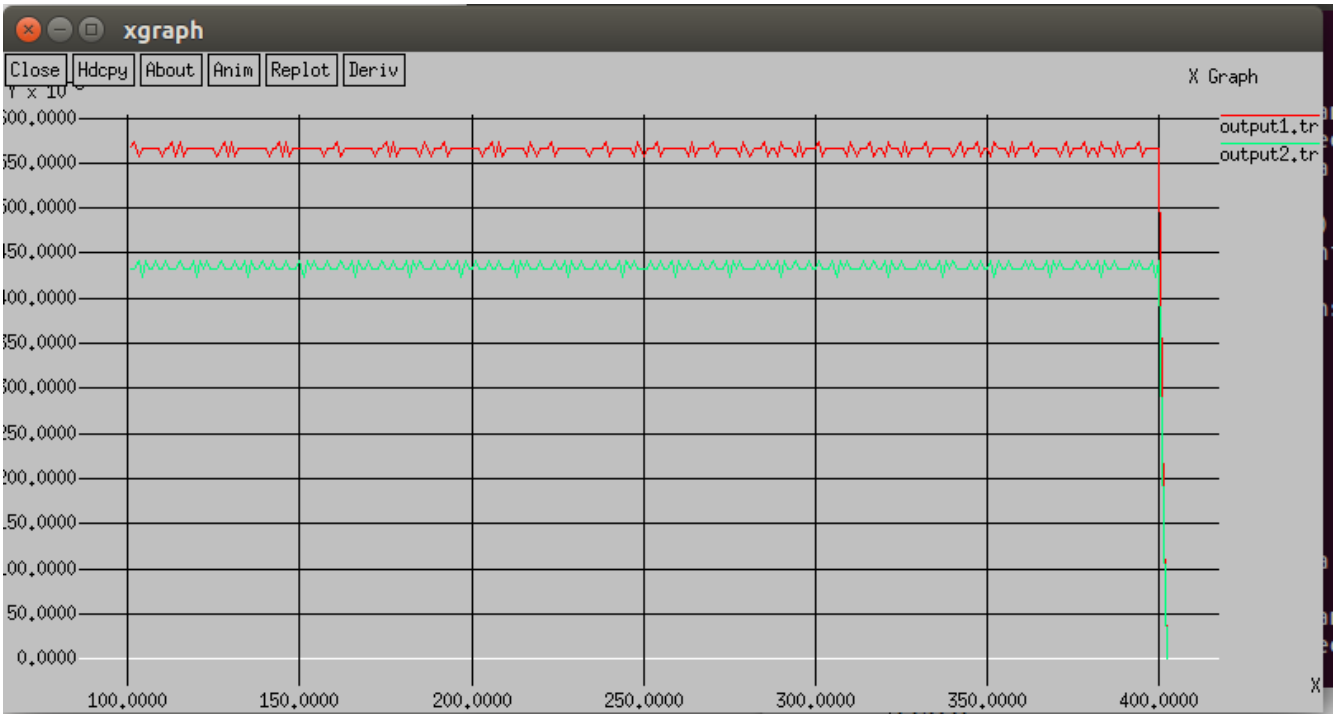
Throughput ratio – 1.1999



c) case 3

FLOW	THROUGHPUT(Mbps)
(src 1 – rcv 1)	0.5652
(src 2 – rcv 2)	0.4347

Throughput ratio – 1.300



Q2. TCP SACK is much better because it doesn't muffle one of the flows. In TCP VEGAS, as delay ratio increases, more packets of src2 are dropped, but it doesn't happen in TCP SACK. In a network, different links with different delays can be present and it is very important to fairly allocate the links. Hence TCP SACK is a much more balance algorithm for this particular network topology.

But in general, TCP VEGAS is a much better algorithm because,

1. TCP VEGAS has a congestion avoidance mechanism as opposed to congestion detection mechanism in TCP SACK
2. TCP SACK uses selective acknowledge mechanism (window) and this requires extra bits in the messages. This results in lesser utilization of bandwidth

In the simulation SACK is better for flow 2 because, window is large and if some packets are dropped, then VEGAS sends the entire window but SACK sends packets selectively. As delay gets larger sending the entire window across and waiting for acks when FIFO tail drop is used becomes inefficient.

Source code for NS2

```
#
#NS2 Simulation
#Group 20 ECEN 602
#Himanshu Gupta, Adarsh
#Network simulation
#


#Setup the simulator for NS
set ns [new Simulator]

#Check version and test case args
if {$argc!=2} {
        puts "Incorrect arguments, USAGE: ns2 <VERSION> <test case>"
        exit 0
}

set version [lindex $argv 0]
set case    [lindex $argv 1]

puts "Testing the TCP version: $version"
puts "Test Case: $case"

#Open the files in write mode
set nf [open output.nam w]
set tf [open output.tr w]

#Enable trace
$ns trace-all $tf
$ns namtrace-all $nf

#Setting up output files for write mode
set output1 [open output1.tr w]
set output2 [open output2.tr w]
set output3 [open output3.tr w]

#Initial TCP Sink values
set initial1 0
```

```
set initial2 0

set sum1 0
set sum2 0


#Record function to calcualte the throughput values.
#Calculates the bytes sent recursively
proc record {} {

        #Global variable declaration
        global ns output1 output2 output3 sink1 sink2 initial1 initial2 sum1 sum2

        #Add up the tcp bytes in sink nodes, used to
        #calculate the total bytes received
        set bw0 [$sink1 set bytes_]
        set bw1 [$sink2 set bytes_]

        #set the time variable
        set time 1.0

        #Current time value
        set now [$ns now]

        if {$now == 100} {

                set initial1 $bw0
                set initial2 $bw1
        }

        #Calculate the output values from t=100
        if {$now > 100 } {

                #this expression evaluates the bandwidth
                set sum1 [expr $sum1 + $bw0/$time*8/1000000]
                set sum2 [expr $sum2 + $bw1/$time*8/1000000]

                set ratio [expr $sum1/$sum2]
```

```
            #this evaluates the current bandwidth and
            #dumps this value to output files
            set b1 [expr $bw0/$time*8/1000000]
            set b2 [expr $bw1/$time*8/1000000]

            #Dump the values to output trace files
            puts $output1 "$now $b1"
            puts $output2 "$now $b2"
            puts $output3 "$ratio"
        }

        if {$now == 400} {
            puts "average throughput for link1  = [expr $sum1 * $time/300] Mbps"

            puts "average throughput for link2  = [expr $sum2 * $time/300] Mbps"
              puts "Total data sent in link 1 = [expr $sum1] Mb"
              puts "Total data sent in link 2 = [expr $sum2] Mb"
            set ratio [expr $sum1/$sum2]
            puts "Throughput ratios of the link =  [expr $ratio]"
       }

        #Resets the bytes to zero
        $sink1 set bytes_ 0
        $sink2 set bytes_ 0

        #This calls the record process recursively
        $ns at [expr $now + $time] "record"
}

#Set the routers, senders and receiver nodes
set R1 [$ns node]
set R2 [$ns node]
set s1 [$ns node]
set s2 [$ns node]
set r1 [$ns node]
set r2 [$ns node]

#Setup the link between Two routers
#The link speed and delay is constant
```

```
$ns duplex-link $R1 $R2 1Mb 5ms DropTail

#Setup the link and delay vlaues according to
# the test case spcified in the argument
if {$case == 1 } {
        $ns duplex-link $s1 $R1 10Mb 5ms DropTail
        $ns duplex-link $R2 $r1 10Mb 5ms DropTail
        $ns duplex-link $s2 $R1 10Mb 12.5ms DropTail
        $ns duplex-link $R2 $r2 10Mb 12.5ms DropTail
} elseif {$case == 2} {
        $ns duplex-link $s1 $R1 10Mb 5ms DropTail
        $ns duplex-link $R2 $r1 10Mb 5ms DropTail
        $ns duplex-link $s2 $R1 10Mb 20ms DropTail
        $ns duplex-link $R2 $r2 10Mb 20ms DropTail
} elseif {$case == 3} {
        $ns duplex-link $s1 $R1 10Mb 5ms DropTail
        $ns duplex-link $R2 $r1 10Mb 5ms DropTail
        $ns duplex-link $s2 $R1 10Mb 27.5ms DropTail
        $ns duplex-link $R2 $r2 10Mb 27.5ms DropTail
}

#Setup the TCP version according the specified version in arguments
if {$version == "SACK"} {
        set tcp1 [new Agent/TCP/Sack1]
        set tcp2 [new Agent/TCP/Sack1]
}

if {$version == "VEGAS"} {
        set tcp1 [new Agent/TCP/Vegas]
        set tcp2 [new Agent/TCP/Vegas]
}

#Setup the TCP sink for receive side
set sink1 [new Agent/TCPSink]
set sink2 [new Agent/TCPSink]

#Attach the tcp source and sink nodes
$ns attach-agent $s1 $tcp1
$ns attach-agent $s2 $tcp2
```

```
$ns attach-agent $r1 $sink1
$ns attach-agent $r2 $sink2

#Connect the source and sink nodes
$ns connect $tcp1 $sink1
$ns connect $tcp2 $sink2

#Create a new FTP application
set ftp1 [new Application/FTP]
set ftp2 [new Application/FTP]

#Attacht the FTP agents to tcp source nodes
$ftp1 attach-agent $tcp1
$ftp2 attach-agent $tcp2

#Create a layout for animation

$R1 shape box
$R2 shape box


$ns duplex-link-op $R1 $R2 orient right
$ns duplex-link-op $s1 $R1 orient 310deg
$ns duplex-link-op $r1 $R2 orient 230deg
$ns duplex-link-op $s2 $R1 orient 50deg
$ns duplex-link-op $r2 $R2 orient 130deg


#Finish function called after 400 sec
proc finish {} {
        global ns nf tf
        $ns flush-trace

        close $nf
        close $tf

        exec xgraph output1.tr output2.tr -geometry 800x400 &
        exec nam output.nam &
        puts "Simlulation finished, use xgraph to plot the results."
```

```
        exit 0
}

#Start the simulation
$ns at 1 "$ftp1 start"
$ns at 1 "$ftp2 start"

#Call Record
$ns at 100 "record"

$ns at 400 "$ftp1 stop"
$ns at 400 "$ftp2 stop"


#Call finish
$ns at 403 "finish"

$ns run
```