**ECEN 602**

**NETWORK SIMULATION ASSIGNMENT**

**TEAM 17**

**Mohammad Faisal Khan**

**Amiya Ranjan Panda**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**README**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**TITLE : NS 2 Familirisation**

**INTRODUCTION :**

**This code is a part of the Network simulation Assignment for ECEN 602 at Texas A&M University.**

**It has been sucessfully compiled, executed and tested on Network Simulator - 2 environment.**

**Ns is a discrete event simulator targeted at networking research. Ns provides substantial support**

**for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite)**

**networks. NS2 consists of two key languages: CCC and Object-oriented Tool Command Language (OTcl).**

**While the CCC defines the internal mechanism (i.e., a backend) of the simulation, the OTcl sets up**

**simulation by assembling and configuring the objects as well as scheduling discrete events**

**(i.e., a frontend). The CCC and the OTcl are linked together using TclCL. After simulation, NS2**

**outputs either text-based simulation results. To interpret these results graphically and interactively,**

**tools such as NAM (Network AniMator) and XGraph are used. To analyze a particular behavior of the network,**

**users can extract a relevant subset of text-based data and transform it to a more conceivable presentation.**

**This step implements the design in the first step. It consists of two phases:**

**• Network configuration phase: In this phase, network components (e.g., node,**

**TCP and UDP) are created and configured according to the simulation design.**

**Also, the events such as data transfer are scheduled to start at a certain**

**time.**

**• Simulation Phase: This phase starts the simulation which was configured in the**

**Network Configuration Phase. It maintains the simulation clock and executes**

**events chronologically. This phase usually runs until the simulation clock reaches**

**a threshold value specified in the Network Configuration Phase.**

**TASK:**

**Use the NS-2 simulator to build the following configuration:**

**• Two routers (R1, R2) connected with a 1 Mbps link and 5ms of latency**

**• Two senders (src1, src2) connected to R1 with 10 Mbps links**

**• Two receivers (rcv1, rcv2) connected to R2 with 10 Mbps links**

**• Application sender is FTP over TCP**

**Run 400s simulations for the following variable parameters:**

**TCP version = (TCP SACK | TCP 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 = 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**

**USAGE:**

**1. open bash window.**

**2. Enter input in the below given format:**

**ns ns2.tcl <tcp type> <case no>**

**wherein, <tcp type> - "vegas" or "sack"**

**<case no> - "1", "2", "3"**

|  |  |  |  |
| --- | --- | --- | --- |
| SACK (THROUGHPUT) | **SRC1 (MBits/sec)** | **SRC2 (MBits/sec)** | **RATIO (SRC1/SRC2)** |
| **CASE 1** | 0.682 | 0.649 | 1.05 |
| **CASE 2** | 0.697 | 0.633 | 1.10 |
| **CASE 3** | 0.712 | 0.618 | 1.15 |

|  |  |  |  |
| --- | --- | --- | --- |
| VEGAS (THROUGHPUT) | **SRC1 (MBits/sec)** | **SRC2 (MBits/sec)** | **RATIO (SRC1/SRC2)** |
| **CASE 1** | 0.665 | 0.665 | 1.0004 |
| **CASE 2** | 0.726 | 0.605 | 1.2 |
| **CASE 3** | 0.776 | 0.554 | 1.4 |

**From the results, the throughput for SRC1 increases and the throughput for SRC2 decreases as the RTT is increased.**

**For case 1, the throughput in SRC1 is more in SACK as compared to VEGAS. The throughput of SRC2 on the contrary is more in case of VEGAS as compared to SACK.**

**The reason for that is because in case of VEGAS, the performance is better at low RTT as in SACK, it requires congestion and acknowledgement from the receiver to transmit data.**

**#Input from user**

**set a [lindex $argv 0];**

**set b [lindex $argv 1];**

**#Initialising netsim**

**set ns [new Simulator];**

**#creating animator file**

**set f [open output.nam w];**

**$ns namtrace-all $f;**

**#creating trace dump file**

**set nf [open test.tr w];**

**$ns trace-all $nf;**

**set throughputsrc1 0**

**set throughputsrc2 0**

**set netcounter 0**

**#defining "end" function**

**proc end {} {**

**global ns f nf throughputsrc1 throughputsrc2 netcounter;**

**$ns flush-trace;**

**puts "Avg throughput for Src1=[expr $throughputsrc1/$netcounter] MBits/sec\n"**

**puts "Avg throughput for Src2=[expr $throughputsrc2/$netcounter] MBits/sec\n"**

**close $f;**

**close $nf;**

**exec nam output.nam &;**

**exit 0;**

**}**

**#defining color coding**

**$ns color 1 Red;**

**$ns color 2 Blue;**

**#defining nodes**

**set R1 [$ns node];**

**set R2 [$ns node];**

**set src1 [$ns node];**

**set src2 [$ns node];**

**set dst1 [$ns node];**

**set dst2 [$ns node];**

**#defining links**

**$ns duplex-link $R1 $R2 1mb 5ms DropTail;**

**$ns duplex-link $src1 $R1 10mb 0ms DropTail;**

**$ns duplex-link $dst1 $R2 10mb 0ms DropTail;**

**if {$b == 1} {**

**$ns duplex-link $src2 $R1 10mb 3.75ms DropTail;**

**$ns duplex-link $dst2 $R2 10mb 3.75ms DropTail;**

**} elseif {$b == 2} {**

**$ns duplex-link $src2 $R1 10mb 7.5ms DropTail;**

**$ns duplex-link $dst2 $R2 10mb 7.5ms DropTail;**

**} elseif {$b == 3} {**

**$ns duplex-link $src2 $R1 10mb 11.25ms DropTail;**

**$ns duplex-link $dst2 $R2 10mb 11.25ms DropTail;**

**}**

**#Defining topology**

**$ns duplex-link-op $R1 $R2 orient right;**

**$ns duplex-link-op $src1 $R1 orient right-down;**

**$ns duplex-link-op $dst1 $R2 orient right-up;**

**$ns duplex-link-op $src2 $R1 orient right-up;**

**$ns duplex-link-op $dst1 $R2 orient right-down;**

**#establishing tcp connections**

**if {$a == "sack"} {**

**set tcp [new Agent/TCP/Sack1];**

**set tcp1 [new Agent/TCP/Sack1];**

**} elseif {$a == "vegas"} {**

**set tcp [new Agent/TCP/Vegas];**

**set tcp1 [new Agent/TCP/Vegas];**

**}**

**$ns attach-agent $src2 $tcp1;**

**$ns attach-agent $src1 $tcp;**

**set tcpsink [new Agent/TCPSink];**

**$ns attach-agent $dst1 $tcpsink;**

**set tcpsink1 [new Agent/TCPSink];**

**$ns attach-agent $dst2 $tcpsink1;**

**$ns connect $tcp $tcpsink;**

**$ns connect $tcp1 $tcpsink1;**

**#establishing ftp connections**

**set ftp [new Application/FTP];**

**$ftp attach-agent $tcp;**

**set ftp1 [new Application/FTP];**

**$ftp1 attach-agent $tcp1;**

**#Defining classes for color coding**

**$tcp set class\_ 1;**

**$tcp1 set class\_ 2;**

**#Recording data in output files**

**set rcd1 [open record\_1.tr w]**

**set rcd2 [open record\_2.tr w]**

**#record procedure to actually write data to output files**

**proc record {} {**

**global tcpsink tcpsink1 rcd1 rcd2 throughputsrc1 throughputsrc2 netcounter**

**#an instance of simulator**

**set ns [Simulator instance]**

**#set time after which procedure will be called again**

**set time 0.5**

**#get current time**

**set now [$ns now]**

**#traffic received by sinks**

**set bw1 [$tcpsink set bytes\_]**

**set bw2 [$tcpsink1 set bytes\_]**

**# calculate the bandwidth (in MBit/s) and write it to the files**

**puts $rcd1 "$now [expr $bw1/$time\*8/1000000]"**

**puts $rcd2 "$now [expr $bw2/$time\*8/1000000]"**

**set throughputsrc1 [expr $throughputsrc1+ $bw1/$time\*8/1000000 ]**

**set throughputsrc2 [expr $throughputsrc2+ $bw2/$time\*8/1000000 ]**

**set netcounter [expr $netcounter + 1]**

**#reset bytes\_ on the sinks**

**$tcpsink set bytes\_ 0**

**$tcpsink1 set bytes\_ 0**

**# re-schedule the procedure**

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

**}**

**#labelling nodes**

**$ns at 0 "$src1 label source1";**

**$ns at 0 "$src2 label source2";**

**$ns at 0 "$R1 label Router1";**

**$ns at 0 "$R2 label Router2";**

**$ns at 0 "$dst1 label receiver1";**

**$ns at 0 "$dst2 label receiver2";**

**#initialising testing**

**$ns at 0 "$ftp start";**

**$ns at 0 "$ftp1 start";**

**$ns at 100 "record"**

**$ns at 400 "$ftp stop";**

**$ns at 400 "$ftp1 stop";**

**$ns at 400 "end";**

**$ns run;**