Introduction

The aim of this project is to run a 400 seconds simulation using ns with different version of TCPs and delay to observe the behaviors and characteristics in each case. TCP versions used in the simulation are VEGAS and SACK.

Set up

Operation system: ubuntu 64 bits on virtual machine.

Tool used: ns version is allinone-2.35

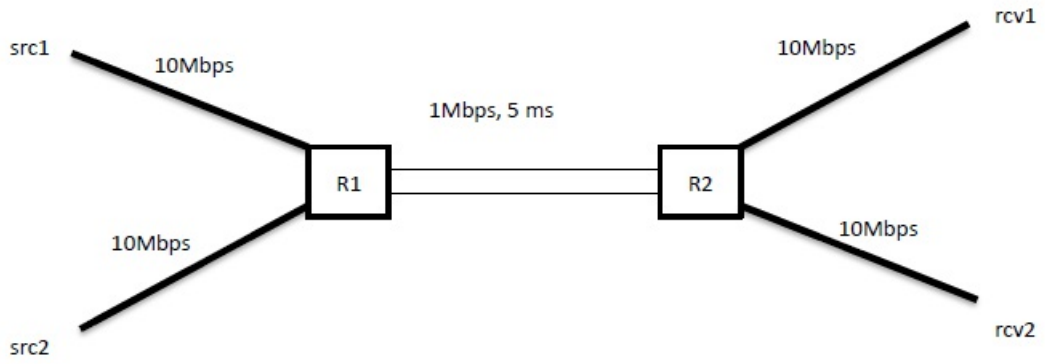
Installnation: cd into the downloaded packet and run ./install

Steps

1. Installnation of ns tools and examine it with testing code.
2. Write TCL scripts to simulate network with given configuration.
3. Result analysis

Configuration

Topology:



Case one:

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

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

Case two:

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

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

Case three:

src-1R1 and R2-rcv1 end to end delay = 5 ms

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

Running command

```
ns ns2.tcl TCP_version case_number
```

TCP version could be VEGAS or SACK. And the case number could be chosen from 1 to 3. Example: ns ns2.tcl VEGAS 2

Result

VEGAS			SACK		
1	output 1	0.583	1	output 1	0.532
	output 2	0.417		output 2	0.466
2	output 1	0.688	2	output 1	0.545
	output 2	0.313		output 2	0.455
3	output 1	0.75	3	output 1	0.565
	output 2	0.25		output 2	0.435

Table 1: Average throughput of different verstions of TCP and configuration

VEGAS		SACK	
Case	output 1/output 2	Case	output 1/output 2
1	1.4	1	1.10
2	2.2	2	1.20
3	3	3	1.31

Table 2: table recording ratio of output one and output two

Analysis

By comparing the same TCP version in different cases, it could be found that the throughput decreases when RTT gets larger. The ratio of output 1 to output 2 becomes larger. It could be found the increase of VEGAS is much more obvious than that of SACK.

By comparing VEGAS and SACK, it could be noticed that in each case, the ratio of out1/out2 of VEGAS is largers than that of SACK. A larger ratio represents a better performance.

We could draw the conclusion that SACK is better than VEGAS.

tcl scripts

```
set ns [new Simulator]

#open nam trace file
```

```
set nf [open output.nam w]
$ns namtrace-all $nf

#take in input parameters
#Report error when input is not correct
if { $argc != 2 } {
        puts "Invalid input!"
    }
set flavor [lindex $argv 0]
set input_case [lindex $argv 1]
if {$input_case > 3 || $input_case < 1} {
        puts "Invalid input case $input_case"
        exit
}
global flav, delay
set delay 0
switch $input_case {
        global delay
        1 {set delay "12.5ms"}
        2 {set delay "20ms"}
        3 {set delay "27.5ms"}
}
if {$flavor == "SACK"} {
        set flav "Sack1"
} elseif {$flavor == "VEGAS"} {
        set flav "Vegas"
} else {
        puts "Invalid TCP Flavor $flavor"
        exit
}

#create 6 nodes
set source1 [$ns node]
set source2 [$ns node]
set router1 [$ns node]
set router2 [$ns node]
set receiver1 [$ns node]
set receiver2 [$ns node]

#create TCP connection
set tcp1 [new Agent/TCP/$flav]
$ns attach-agent $source1 $tcp1
set tcp2 [new Agent/TCP/$flav]
$ns attach-agent $source2 $tcp2

#create links between nodes with the input delay
$ns duplex-link $source1 $router1 10Mb 5ms DropTail
$ns duplex-link $source2 $router1 10Mb $delay DropTail
$ns duplex-link $router1 $router2 1Mb 5ms DropTail
$ns duplex-link $router2 $receiver1 10Mb 5ms DropTail
$ns duplex-link $router2 $receiver2 10Mb $delay DropTail

$ns duplex-link-op $source1 $router1 orient right-down
$ns duplex-link-op $source2 $router1 orient right-up
$ns duplex-link-op $router1 $router2 orient right
$ns duplex-link-op $router2 $receiver1 orient right-up
$ns duplex-link-op $router2 $receiver2 orient right-down


set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1

set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
```

```
set null1 [new Agent/TCPSink]
$ns attach-agent $receiver1 $null1

set null2 [new Agent/TCPSink]
$ns attach-agent $receiver2 $null2


$ns connect $tcp1 $null1
$ns connect $tcp2 $null2

#schedule events
#starts recording at 0 and finish at 400
$ns at 0 "record"
$ns at 0 "$ftp1 start"
$ns at 400 "$ftp1 stop"

$ns at 0 "$ftp2 start"
$ns at 400 "$ftp2 stop"

$ns at 400 "finish"

set testfile1 [open "file-S1.tr" w]
$ns trace-queue  $source1  $router1  $testfile1

set testfile2 [open "file-S2.tr" w]
$ns trace-queue  $source2  $router1  $testfile2
proc finish {} {
    global ns nf sum1 sum2 cnt
        $ns flush-trace
        close $nf
        exec nam -a output.nam &
        puts "Avgerage throughput for source1=[expr $sum1/$cnt] MBits/sec\n"
        puts "Avgerage throughput for source2=[expr $sum2/$cnt] MBits/sec\n"
    exit 0
}

set file1 [open out1.tr w]
set file2 [open out2.tr w]

proc record {} {
        global null1 null2 file1 file2 sum1 sum2 cnt
        set ns [Simulator instance]
        set time 0.5
                set cnt 0
                set sum1 0
                set sum2 0
        set bw1 [$null1 set bytes_]
        set bw2 [$null2 set bytes_]
        set now [$ns now]
        puts $file1 "$now [expr $bw1/$time*8/1000000]"
        puts $file2 "$now [expr $bw2/$time*8/1000000]"
                set sum1 [expr $sum1 + $bw1/$time*8/1000000]
                set sum2 [expr $sum2 + $bw2/$time*8/1000000]
                set cnt [expr $cnt+1]
        #Reset the bytes_ values on sinks
        $null1 set bytes_ 0
        $null2 set bytes_ 0
        #Re-schedule the procedure
        $ns at [expr $now+$time] "record"
}

$ns run
```