5th Group Homework Report
NS-2 Familiarization
ECEN 602 Network Simulation Assignment 1

Mainly, we worked together, but here is the role what we did.
Minhwan Oh : Developed client programming, tested for integration
Sanghyeon Lee : Developed server programming, debugged for integration

File info
==========
For this program, you can see how NS-2 simulator works
1. ns2.tcl
Main feature: NS-2 simulation source code

Build info
==========
NS2 install reference: https://www.youtube.com/watch?v=nIdWtN_PwkM
Command line: $ns ns2.tcl <TCP flavor> <Case number>

Simulation scenario
==========

1. Using NS-2 simulator, build the following configuration
- Two routers (R1, R2) connected with a 1Mbps 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

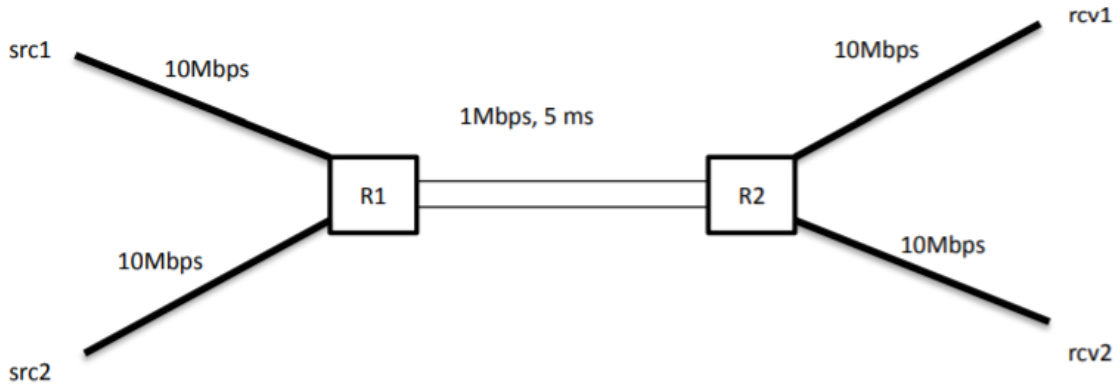
2. Run 400s simulations for the following variable parameters
- TCP version = (TCP SACK | TCP VEGAS)
- Case 1:
  : src1-R1 and R2-rcv1 end-2-end delay = 5 ms
  : src2-R1 and R2-rcv2 end-2-end delay = 12.5 ms
- Case 2:
  : src1-R1 and R2-rcv1 end-2-end delay = 5 ms
  : src2-R1 and R2-rcv2 end-2-end delay = 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

Note that in the cases above the end-to-end RTTs of the two sources are in the ratio 1:2, 1:3 and 1:4, respectively

3. Give the answers to the below requests
(i) For each of the TCP flavors (VEGAS and SACK) simulate the three RTT cases and find the ratio of the average throughput of src1 to src2. Make two separate tables (one for each TCP flavor) showing the throughput for each test case.
(ii) Discuss the relationship between TCP throughput and RTT in light of your results in (i) above for each TCP flavor. Also compare and discuss the throughput performance of the two flavors of TCP for Case 1. Explain why one of the TCP flavors performs better than the other for Case 1. There is information about TCP Vegas in our textbook, pp. 523-530. RFC 6675 describes the current recommended TCP SACK operation, which is basically TCP RENO + SACK. I am not sure if ns-2 implements RFC 6675 or the earlier RFC 3517 that was made obsolete by RFC 6675, however.

Note: Run the simulations for 400 seconds, and ignore the first 100 seconds while measuring metrics.

Program details & File architecture
==========
1. Simulation design



- Using OTcl and C++ script, describe the TCP network
- After 100s from beginning, it measures the RTT and check its average throughput to see their ratio.
- In simulation code, we made new simulator, and made trace value for visualization through simulator.
- Then, we designed the network satisfying above conditions by defining noes, links, and queuing.
- We can make VEGAS and SACK TCP versions by setting ns options via OTcl code.
- We set senders and receivers color depend on its case options; case 1: is red, case 2 is green, and case 3 is blue.
- out.nam stores the traces so that we can visually see it on NS simulator.

File Function Explanation
==========
1. ns2.tcl
a) Architecture
- Scenario :
(1) Setup TCP networks
(2) Track the outputs to visualize it on NS simulator

TEST CASE

(i) For each of the TCP flavors (VEGAS and SACK) simulate the three RTT cases and find the ratio of the average throughput of src1 to src2. Make two separate tables (one for each TCP flavor) showing the throughput for each test case.

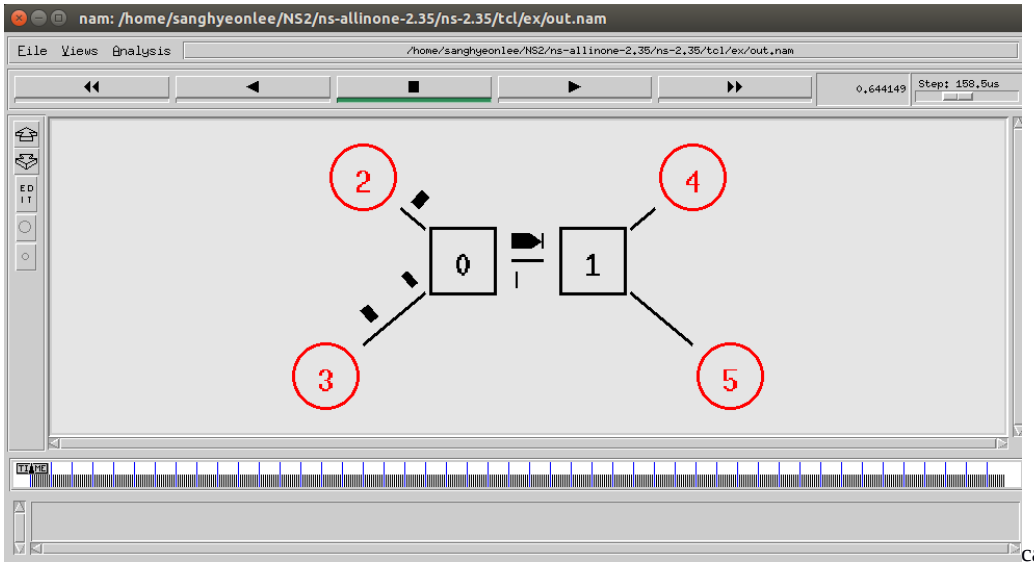
	Average throughput ratio between src1 and src2		
TCP flavor	CASE 1	CASE 2	CASE 3
VEGAS	1 : 1.4	1 : 2.2	1 : 3

	Average throughput ratio between src1 and src2		
TCP flavor	CASE 1	CASE 2	CASE 3
SACK	1 : 1.1	1 : 1.19	1 : 1.3

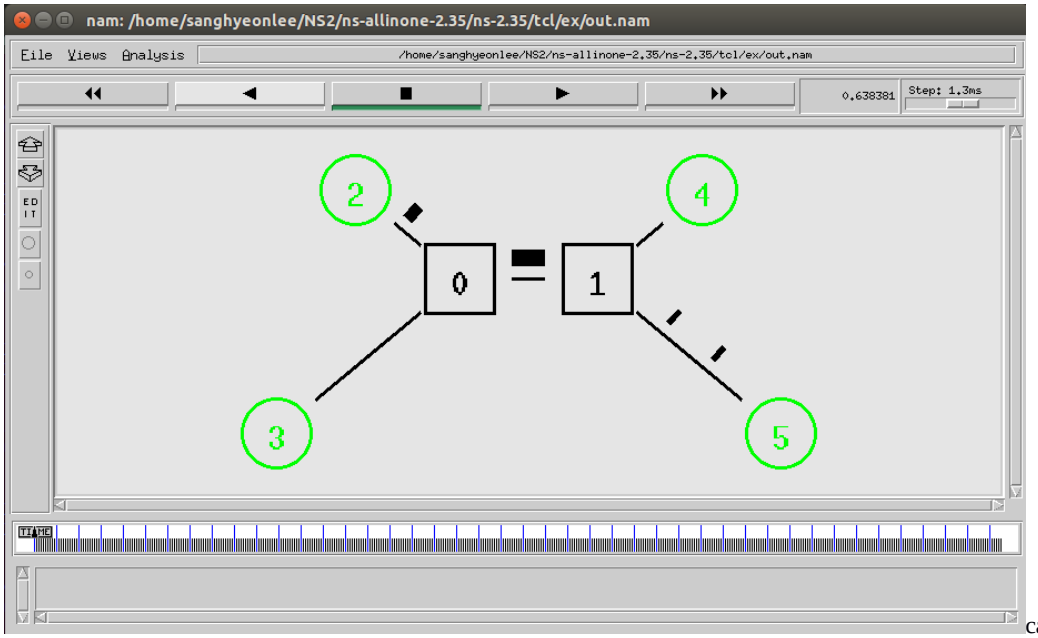
(ii) Discuss the relationship between TCP throughput and RTT in light of your results in (i) above for each TCP flavor. Also compare and discuss the throughput performance of the two flavors of TCP for Case 1. Explain why one of the TCP flavors performs better than the other for Case 1. There is information about TCP Vegas in our textbook, pp. 523-530. RFC 6675 describes the current recommended TCP SACK operation, which is basically TCP RENO + SACK. I am not sure if ns-2 implements RFC 6675 or the earlier RFC 3517 that was made obsolete by RFC 6675, however.

When RTT is increased, TCP throughput is also increased. This is because while the delay is increased, the source can get more packets due to the increased time period. In addition, as we can see from above table, the average throughput ratio of VEGAS is increasing higher than those of SACK. By observing throughput ratio, we can check the congestion more effectively. Thus, TCP VEGAS has better bandwidth and less congestion.

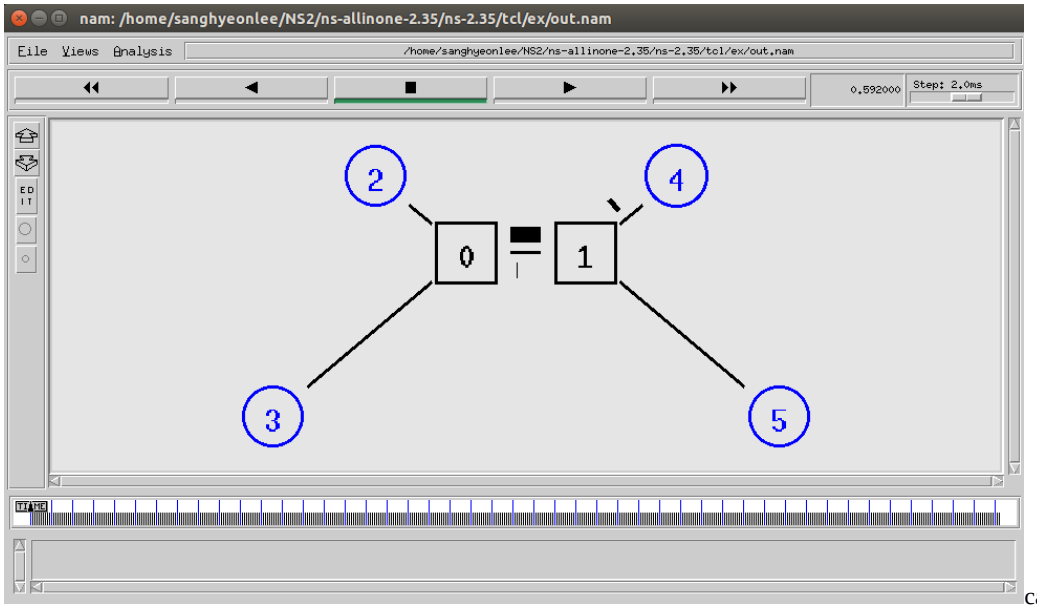
```
sanghyeonlee@sanghyeonlee-VirtualBox: ~/NS2/ns-allinone-2.35/ns-2.35/tcl/ex
sanghyeonlee@sanghyeonlee-VirtualBox:~/NS2/ns-allinone-2.35/ns-2.35/tcl/ex$ ns ns2.tcl VEGAS 1
TCP flavor : VEGAS
Case 1:
                - src1-R1 and R2-rcv1 end-2-end delay = 5 ms
                - src2-R1 and R2-rcv2 end-2-end delay = 12.5 ms
                - RTT ratio 1:2
The ratio of  the average throughput of src1 to src2
                0.58333333333333459 : 0.41666666666666435 = 1 : 1.4000000000000108
sanghyeonlee@sanghyeonlee-VirtualBox:~/NS2/ns-allinone-2.35/ns-2.35/tcl/ex$ ns ns2.tcl VEGAS 2
TCP flavor : VEGAS
Case 2:
                - src1-R1 and R2-rcv1 end-2-end delay = 5 ms
                - src2-R1 and R2-rcv2 end-2-end delay = 20 ms
                - RTT ratio 1:3
The ratio of  the average throughput of src1 to src2
                0.68751999999999713 : 0.3125066666666651 = 1 : 2.2000170663025873
sanghyeonlee@sanghyeonlee-VirtualBox:~/NS2/ns-allinone-2.35/ns-2.35/tcl/ex$ ns ns2.tcl VEGAS 3
TCP flavor : VEGAS
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
                - RTT ratio 1:4
The ratio of  the average throughput of src1 to src2
                0.75000000000000078 : 0.24999999999999992 = 1 : 3.000000000000004
sanghyeonlee@sanghyeonlee-VirtualBox:~/NS2/ns-allinone-2.35/ns-2.35/tcl/ex$ ns ns2.tcl SACK 1
TCP flavor : 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
                - RTT ratio 1:2
The ratio of  the average throughput of src1 to src2
                0.52382720000000171 : 0.47618133333333501 = 1 : 1.1000582411182291
sanghyeonlee@sanghyeonlee-VirtualBox:~/NS2/ns-allinone-2.35/ns-2.35/tcl/ex$ ns ns2.tcl SACK 2
TCP flavor : SACK
Case 2:
                - src1-R1 and R2-rcv1 end-2-end delay = 5 ms
                - src2-R1 and R2-rcv2 end-2-end delay = 20 ms
                - RTT ratio 1:3
The ratio of  the average throughput of src1 to src2
                0.54543146666666775 : 0.4545770666666642 = 1 : 1.1998657800012291
sanghyeonlee@sanghyeonlee-VirtualBox:~/NS2/ns-allinone-2.35/ns-2.35/tcl/ex$ ns ns2.tcl SACK 3
TCP flavor : SACK
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
                - RTT ratio 1:4
The ratio of  the average throughput of src1 to src2
                0.56523306666667206 : 0.43477546666666345 = 1 : 1.3000574089430597
sanghyeonlee@sanghyeonlee-VirtualBox:~/NS2/ns-allinone-2.35/ns-2.35/tcl/ex$
```



case 1



case 2



case 3

Ns2.tcl

```
#New simulator Object
set ns [new Simulator]


#Check Input Argument
if {$argc!=2} {
        puts "[System] : Argument Setting Error"
        puts "                  Set Argument : ns ns2.tcl <TCP_flavor> <case_no>"
        puts "                  Example: ns ns2.tcl VEGAS 2"
        exit 0
}

#Set Output
set tf [open out.tr w]
set output1 [open src1_output.tr w]
set output2 [open src2_output.tr w]
set output3 [open ratio_output.tr w]

set nf [open out.nam w]

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

#Initialize variables
set sum1 0
set sum2 0
set counter 0

#Calculate Throughput
proc calculate {} {

        #Variables
        global ns sink1 sink2 output1 output2 output3 sum1 sum2 counter

        set bw0 [$sink1 set bytes_]
        set bw1 [$sink2 set bytes_]


        #Step time
        set time 0.5

        #Current time
        set now [$ns now]


        #Ignore the first 100 seconds
        if {$now == 100} {
                $sink1 set bytes_ 0
                $sink2 set bytes_ 0
        }

        #Measure Throughput from 100 to 400 seconds
        if {$now > 100 && $now<= 400 } {
                set throughput1 [expr $bw0/$time *8/1000000]
                set throughput2 [expr $bw1/$time *8/1000000]
                set ratio [expr $throughput1/$throughput2]

                set sum1 [expr $sum1 + $throughput1]
                set sum2 [expr $sum2 + $throughput2]
                set counter [expr $counter + 1]

                #Store output values
                puts $output1 "$now $throughput1"
                puts $output2 "$now $throughput2"
```

```
                        puts $output3 "$ratio"

                        $sink1 set bytes_ 0
                        $sink2 set bytes_ 0
                }

                if { $now == 400.5 } {
                        set avgthroughput1 [ expr $sum1/$counter]
                        set avgthroughput2 [ expr $sum2/$counter]
                        set ratio [expr $avgthroughput1/$avgthroughput2]
                        puts "The ratio of  the average throughput of src1 to src2 "
                        puts "                      $avgthroughput1 : $avgthroughput2 = 1 : $ratio"
                }

                #Recursion call
                $ns at [expr $now + $time] "calculate"
}



#Set nodes
set R1 [$ns node]
set R2 [$ns node]
set src1 [$ns node]
set src2 [$ns node]
set rcv1 [$ns node]
set rcv2 [$ns node]

#Set Drop Tail
$ns duplex-link $R1 $R2 1Mb 5ms DropTail

#Select TCP flavor
#VEGAS
if {[lindex $argv 0] eq "VEGAS"} {
                puts "TCP flavor : VEGAS"
                set tcp1 [new Agent/TCP/Vegas]
                set tcp2 [new Agent/TCP/Vegas]
}
#SACK
if {[lindex $argv 0] eq "SACK"} {
                puts "TCP flavor : SACK"
                set tcp1 [new Agent/TCP/Sack1]
                set tcp2 [new Agent/TCP/Sack1]
}

#Select CASE
#Case 1 : Red
if {[lindex $argv 1] eq "1"} {
                puts "Case 1:"
                puts "                  - src1-R1 and R2-rcv1 end-2-end delay = 5 ms"
                puts "                  - src2-R1 and R2-rcv2 end-2-end delay = 12.5 ms"
                puts "                  - RTT ratio 1:2"
                $rcv1 color Red
                $rcv2 color Red
                $src1 color Red
                $src2 color Red

                $ns duplex-link $src1 $R1 10Mb 5ms DropTail
                $ns duplex-link $R2 $rcv1 10Mb 5ms DropTail

                $ns duplex-link $src2 $R1 10Mb 12.5ms DropTail
                $ns duplex-link $R2 $rcv2 10Mb 12.5ms DropTail

}

#Case 2 : Green
```

```
if {[lindex $argv 1] eq "2"} {
        puts "Case 2:"
        puts "                  - src1-R1 and R2-rcv1 end-2-end delay = 5 ms"
        puts "                  - src2-R1 and R2-rcv2 end-2-end delay = 20 ms"
        puts "                  - RTT ratio 1:3"

        $rcv1 color Green
        $rcv2 color Green
        $src1 color Green
        $src2 color Green

        $ns duplex-link $src1 $R1 10Mb 5ms DropTail
        $ns duplex-link $R2 $rcv1 10Mb 5ms DropTail

        $ns duplex-link $src2 $R1 10Mb 20ms DropTail
        $ns duplex-link $R2 $rcv2 10Mb 20ms DropTail

}

#Case 3 : Blue
if {[lindex $argv 1] eq "3"} {
        puts "Case 3:"
        puts "                  - src1-R1 and R2-rcv1 end-2-end delay = 5 ms"
        puts "                  - src2-R1 and R2-rcv2 end-2-end delay = 27.5 ms"
        puts "                  - RTT ratio 1:4"

        $rcv1 color Blue
        $rcv2 color Blue
        $src1 color Blue
        $src2 color Blue

        $ns duplex-link $src1 $R1 10Mb 5ms DropTail
        $ns duplex-link $R2 $rcv1 10Mb 5ms DropTail

        $ns duplex-link $src2 $R1 10Mb 27.5ms DropTail
        $ns duplex-link $R2 $rcv2 10Mb 27.5ms DropTail
}



#TCP Sinks initialization
set sink1 [new Agent/TCPSink]
set sink2 [new Agent/TCPSink]

#Attach
$ns attach-agent $src1 $tcp1
$ns attach-agent $src2 $tcp2
$ns attach-agent $rcv1 $sink1
$ns attach-agent $rcv2 $sink2

#Connect
$ns connect $tcp1 $sink1
$ns connect $tcp2 $sink2

#Application sender is FTP over TCP
set ftp1 [new Application/FTP]
set ftp2 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ftp2 attach-agent $tcp2



#Structure
$R1 shape square
$R2 shape square
$R1 color black
$R2 color black
```

```
$ns duplex-link-op $R1 $R2 orient right
$ns duplex-link-op $src1 $R1 orient 320deg
$ns duplex-link-op $src2 $R1 orient 40deg

$ns duplex-link-op $rcv1 $R2 orient 220deg
$ns duplex-link-op $rcv2 $R2 orient 140deg

#Close simulation
proc Complete {} {

        global ns nf tf
        $ns flush-trace

        close $nf
        close $tf

        #exec <nam folder> out.nam &
        exit 0
}

#MAIN----------------------------------------
#Start
$ns at 0.5 "$ftp1 start"
$ns at 0.5 "$ftp2 start"

#Measurement
$ns at 100 "calculate"

$ns at 401 "$ftp1 stop"
$ns at 401 "$ftp2 stop"

#Close
$ns at 405 "Complete"
$ns run
```