HW4-NS2 Familiarization

Tharun Battula, Guoyu Fu

1. Summary:

In the coding assignment, we successfully finished the simulation coding in ns-2 environment; the ns2.tcl code can configure with the required networks and simulate for 400s perfectly; the “awk” function in .tcl code does the text parsing in order to calculate the throughput of src1 and src2, as required.

2. Description:

We wrote Tcl codes that runs with ns-2 tool. The code is attached to the report in appendix. To run the code, simply open the terminal in the directory, type in command “ns ns2.tcl <TCP_flavor> <case_no>”. Fig 1 is given as examples showing how to use our code.

```
tharun@tharun-HP-ProBook-6560b:~/Desktop/602/HW4/ns2$ ns ns2.tcl VEGAS 1
Given TCP_flavor =  VEGAS
Given case_no    =  1
case selected = 1
VEGAS it is
average throughput for link1  = 0.5833333333333347 Mbps
average throughput for link2  = 0.41666666666666552 Mbps
through put ration link1/link2 =  1.4000000000000072
running nam...
tharun@tharun-HP-ProBook-6560b:~/Desktop/602/HW4/ns2$ ns ns2.tcl VEGAS 2
Given TCP_flavor =  VEGAS
Given case_no    =  2
case selected = 2
VEGAS it is
average throughput for link1  = 0.6874933333333304 Mbps
average throughput for link2  = 0.31250666666666521 Mbps
through put ration link1/link2 =  2.1999317347896583
running nam...
tharun@tharun-HP-ProBook-6560b:~/Desktop/602/HW4/ns2$
```

Figure 1: Examples of execution of our code

The ns2.tcl code basically built a network as described in hw4 requirement, ran simulation for 400s, outputted all traces as tcp_0.tr; and “awk” function inside did text parsing to extract useful data out as “throughput.data” which was written with calculated throughputs from 100s to 400s. The simulated network structure depicted by nam is given by Fig 2.

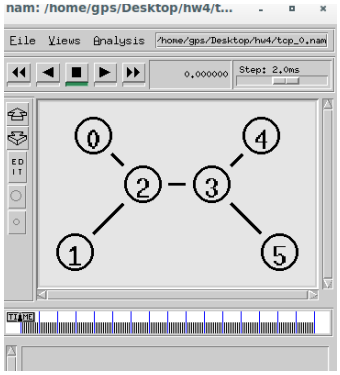

Figure 2: Simulated network in nam

3. Examples:

Example entries in the nam file are given as:

```
r -t 0.0058 -s 0 -d 2 -p tcp -e 1000 -c 0 -i 0 -a 0 -x {0.0 4.0 0 ------- null}
+ -t 0.0058 -s 2 -d 3 -p tcp -e 1000 -c 0 -i 0 -a 0 -x {0.0 4.0 0 ------- null}
- -t 0.0058 -s 2 -d 3 -p tcp -e 1000 -c 0 -i 0 -a 0 -x {0.0 4.0 0 ------- null}
h -t 0.0058 -s 2 -d 3 -p tcp -e 1000 -c 0 -i 0 -a 0 -x {0.0 4.0 -1 ------- null}
r -t 0.0188 -s 2 -d 3 -p tcp -e 1000 -c 0 -i 0 -a 0 -x {0.0 4.0 0 ------- null}
+ -t 0.0188 -s 3 -d 4 -p tcp -e 1000 -c 0 -i 0 -a 0 -x {0.0 4.0 0 ------- null}
- -t 0.0188 -s 3 -d 4 -p tcp -e 1000 -c 0 -i 0 -a 0 -x {0.0 4.0 0 ------- null}
h -t 0.0188 -s 3 -d 4 -p tcp -e 1000 -c 0 -i 0 -a 0 -x {0.0 4.0 -1 ------- null}
r -t 0.0246 -s 3 -d 4 -p tcp -e 1000 -c 0 -i 0 -a 0 -x {0.0 4.0 0 ------- null}
+ -t 0.0246 -s 4 -d 3 -p ack -e 40 -c 0 -i 2 -a 0 -x {4.0 0.0 0 ------- null}
- -t 0.0246 -s 4 -d 3 -p ack -e 40 -c 0 -i 2 -a 0 -x {4.0 0.0 0 ------- null}
h -t 0.0246 -s 4 -d 3 -p ack -e 40 -c 0 -i 2 -a 0 -x {4.0 0.0 -1 ------- null}
r -t 0.0283 -s 1 -d 2 -p tcp -e 1000 -c 1 -i 1 -a 1 -x {1.0 5.0 0 ------- null}
+ -t 0.0283 -s 2 -d 3 -p tcp -e 1000 -c 1 -i 1 -a 1 -x {1.0 5.0 0 ------- null}
- -t 0.0283 -s 2 -d 3 -p tcp -e 1000 -c 1 -i 1 -a 1 -x {1.0 5.0 0 ------- null}
h -t 0.0283 -s 2 -d 3 -p tcp -e 1000 -c 1 -i 1 -a 1 -x {1.0 5.0 -1 ------- null}
r -t 0.029632 -s 4 -d 3 -p ack -e 40 -c 0 -i 2 -a 0 -x {4.0 0.0 0 ------- null}
+ -t 0.029632 -s 3 -d 2 -p ack -e 40 -c 0 -i 2 -a 0 -x {4.0 0.0 0 ------- null}
```

Example entries in the “tcp_o.tr” file are shown below:

```
r 0.175936 2 0 ack 40 ------- 0 4.0 0.0 7 20
+ 0.175936 0 2 tcp 1000 ------- 0 0.0 4.0 12 23
- 0.175936 0 2 tcp 1000 ------- 0 0.0 4.0 12 23
+ 0.175936 0 2 tcp 1000 ------- 0 0.0 4.0 13 24
- 0.176736 0 2 tcp 1000 ------- 0 0.0 4.0 13 24
r 0.179284 2 3 tcp 1000 ------- 0 0.0 4.0 8 18
+ 0.179284 3 4 tcp 1000 ------- 0 0.0 4.0 8 18
- 0.179284 3 4 tcp 1000 ------- 0 0.0 4.0 8 18
r 0.181736 0 2 tcp 1000 ------- 0 0.0 4.0 12 23
+ 0.181736 2 3 tcp 1000 ------- 0 0.0 4.0 12 23
- 0.182284 2 3 tcp 1000 ------- 0 0.0 4.0 10 21
r 0.182536 0 2 tcp 1000 ------- 0 0.0 4.0 13 24
+ 0.182536 2 3 tcp 1000 ------- 0 0.0 4.0 13 24
r 0.185084 3 4 tcp 1000 ------- 0 0.0 4.0 8 18
+ 0.185084 4 3 ack 40 ------- 0 4.0 0.0 8 25
```

4. Results:

The throughputs and the ratio of Src1 to Src2 are given in the following table (Unit: Mbps).

Ratio of RTT	Throughput for TCP SACK				Throughput for TCP VEGAS			
	Src1	Src2	Src1/Src2	Src1+Src2	Src1	Src2	Src1/Src2	Src1+Src2
1:2	0.175	0.6852	0.262	0.8602	0.5833	0.4167	1.4	1
1:3	0.365	0.5041	0.729	0.8691	0.6875	0.3125	2.2	1
1:4	0.128	0.8714	0.147	0.9994	0.6842	0.3158	2.17	1

5. Analysis:

a. SACK means selective acknowledgement policy is used in TCP connection, while VEGAS means congestion avoidance algorithm is used in TCP connection.

b. Speaking of total throughput in the network, VEGAS flavor outperforms SACK obviously, based on the values of “Src1+Src2”. VEGAS can adjust the transmission loads in the network adaptively, so that the network bandwidth (1 Mbps) is fully utilized. Because SACK emphasizes on packet loss rate instead of fully utilization of the network, the flow in SACK fashion will not perfectly match the limit of bandwidth.

c. Speaking of fairness that the throughput should be decided by how many resources (RTT) that each src has, VEGAS also performs better, based on the values of “Src1/Src2”. For case 1, the Src1 has shorter RTT than Src2, so Src1 should have larger throughput than Src2 intuitively. However, in SACK flavor, the Src1’s throughput is only 0.284 of Src2’s, because the ACKs back to Src1 have to wait in queue for depletion of the middle bridge while in the meantime Src1 exhausts the sending window.

Appendix – ns2.tcl

```
#!/usr/bin/awk
set ns [new Simulator]

$ns color 0 blue
$ns color 1 red

set src1 [$ns node]
set src2 [$ns node]
set r1   [$ns node]
set r2   [$ns node]
set rcv1 [$ns node]
set rcv2 [$ns node]

# Initialize variables
set sum_0 0
set sum_1 0
set rat 0

#  src1          rcv1
#         r1-----r2
#  src2          rcv2

set f [open tcp_0.tr w]
$ns trace-all $f
set nf [open tcp_0.nam w]
$ns namtrace-all $nf

if { $argc != 2 } {
      puts "Follow below syntax, Please try again."
         puts "\t\tns2.tcl <TCP_flavor> <case_no>\t\t"
   } else {
         set TCP_flavor [lindex $argv 0]
         set case_no    [lindex $argv 1]
      puts "Given TCP_flavor =  $TCP_flavor"
         puts "Given case_no    =  $case_no"
      }



$ns duplex-link $r1 $r2 1Mb  5ms DropTail
$ns queue-limit $r1 $r2 10

if {$case_no == 1} {
         $ns duplex-link $src1 $r1 10Mb 5ms DropTail
         $ns duplex-link $src2 $r1 10Mb 12.5ms DropTail
         $ns duplex-link $r2 $rcv1 10Mb 5ms DropTail
```

```
        $ns duplex-link $r2 $rcv2 10Mb 12.5ms DropTail
} elseif {$case_no == 2} {
        $ns duplex-link $src1 $r1 10Mb 5ms DropTail
        $ns duplex-link $src2 $r1 10Mb 20ms DropTail
        $ns duplex-link $r2 $rcv1 10Mb 5ms DropTail
        $ns duplex-link $r2 $rcv2 10Mb 20ms DropTail
} elseif {$case_no == 3} {
        $ns duplex-link $src1 $r1 10Mb 5ms DropTail
        $ns duplex-link $src2 $r1 10Mb 27.5ms DropTail
        $ns duplex-link $r2 $rcv1 10Mb 5ms DropTail
        $ns duplex-link $r2 $rcv2 10Mb 27.5ms DropTail
} else {
        puts "Case Not Supported"
}

puts    "case selected = $case_no"

$ns duplex-link-op $src1 $r1 orient right-down
$ns duplex-link-op $src2 $r1 orient right-up
$ns duplex-link-op $r1          $r2 orient right
$ns duplex-link-op $rcv1 $r2 orient left-down
$ns duplex-link-op $rcv2 $r2 orient left-up

$ns duplex-link-op $r1 $r2   queuePos 0.5
$ns duplex-link-op $r2 $rcv1 queuePos 0.5
$ns duplex-link-op $r2 $rcv2 queuePos 0.5


if { $TCP_flavor == "VEGAS"} {
                set tcp_0 [new Agent/TCP/Vegas]
                set tcp_1 [new Agent/TCP/Vegas]
puts "VEGAS it is"
} elseif { $TCP_flavor == "SACK"} {
                set tcp_0 [new Agent/TCP/Sack1]
                set tcp_1 [new Agent/TCP/Sack1]
puts "SACK it is"
} else {
puts "Incorrect TCP flavor"
}

$tcp_0 set class_ 0
$tcp_1 set class_ 1

set sink_0 [new Agent/TCPSink]
$ns attach-agent $src1 $tcp_0
$ns attach-agent $rcv1 $sink_0

$ns connect $tcp_0 $sink_0
# Equivelant of
```

```
# set tcp_0 [$ns create-connection TCP $src1 TCPSink $rcv1 0]

set sink_1 [new Agent/TCPSink]
$ns attach-agent $src2 $tcp_1
$ns attach-agent $rcv2 $sink_1

$ns connect $tcp_1 $sink_1
# Equivelant of
# set tcp_1 [$ns create-connection TCP $src2 TCPSink $rcv2 1]

set ftp_0 [new Application/FTP]
set ftp_1 [new Application/FTP]

$ftp_0 attach-agent $tcp_0
$ftp_1 attach-agent $tcp_1

$ns at   0.0 "record"
$ns at   1.0 "$ftp_0 start"
$ns at   1.0 "$ftp_1 start"
$ns at 401.0 "$ftp_1 stop"
$ns at 401.0 "$ftp_0 stop"
$ns at 402.0 "finish"


proc record {} {
       global sink_0 sink_1 sum_0 sum_1 rat
          set ns [Simulator instance]
          set time 1.0
          set bw_0 [$sink_0 set bytes_]
          set bw_1 [$sink_1 set bytes_]
          set now [$ns now]
          #puts "$now-->\t[expr $bw_0/$time*8/1000000]  [expr $bw_1/$time*8/1000000]"
          if {$now >= 100 && $now < 400 }  {
                    set sum_0 [expr $sum_0+$bw_0/$time*8/1000000]
                    set sum_1 [expr $sum_1+$bw_1/$time*8/1000000]
          }
          if {$now == 400} {
                    puts "average throughput for link1  = [expr $sum_0 * $time/300] Mbps"
                    puts "average throughput for link2  = [expr $sum_1 * $time/300] Mbps"
                    set rat [expr $sum_0/$sum_1]
                    puts "through put ration link1/link2 =  [expr $rat]"
          }
          $sink_0 set bytes_ 0
          $sink_1 set bytes_ 0
          $ns at [expr $now+$time] "record"
}


proc finish {} {
```

```
        global ns f nf
        global ns data
        $ns flush-trace
        close $nf
        close $f
        puts "running nam..."
        exec nam tcp_0.nam &
        #exec xgraph throughput.data &
        exit 0
}

$ns run
```