NS2 Simulation Report

Group #30

Jiaju Shi

Raghavan S V

Hari Kishan Srikonda

Setup:

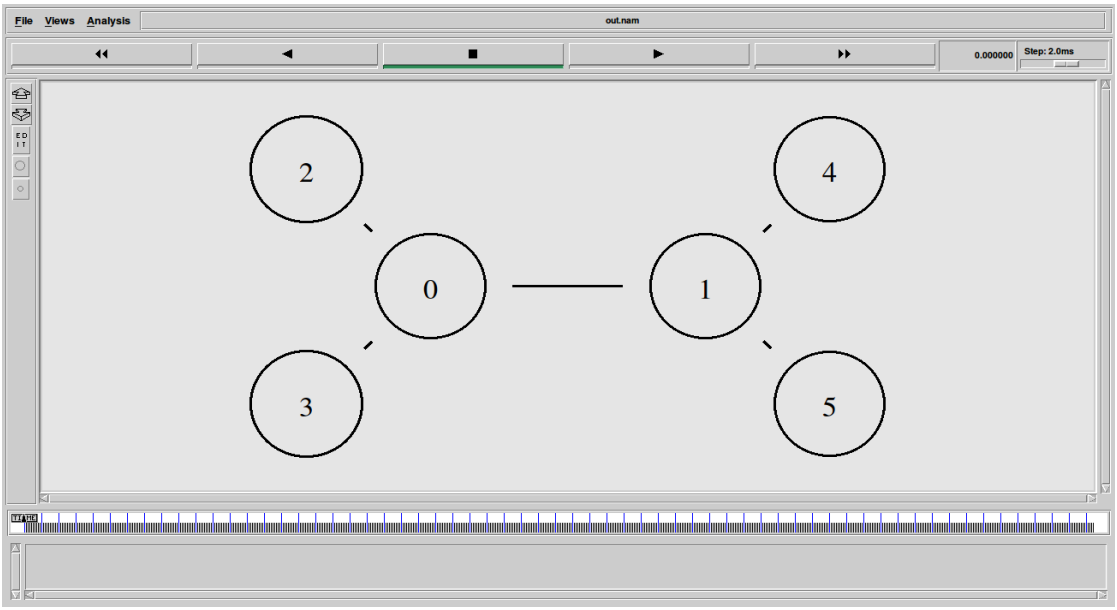
Ubuntu 14.04 (64-bit), NS-2, Xgraph, Nam, Tcl/Tk

Procedure:

First, we install ns2 (Network Simulator – 2), NAM (network animator) and Xgraph packages on Ubuntu. First we start simulation and record data at time 0.0, and at time 30.0, we call clear() procedure to clear all the data. During time 30.0~180.0, we call record procedure to calculate the instantaneous throughput for time interval 1 s. At time 180.0, we stop simulation and call finish procedure, and we will plot the graph at the finish procedure.

Scenario 1:

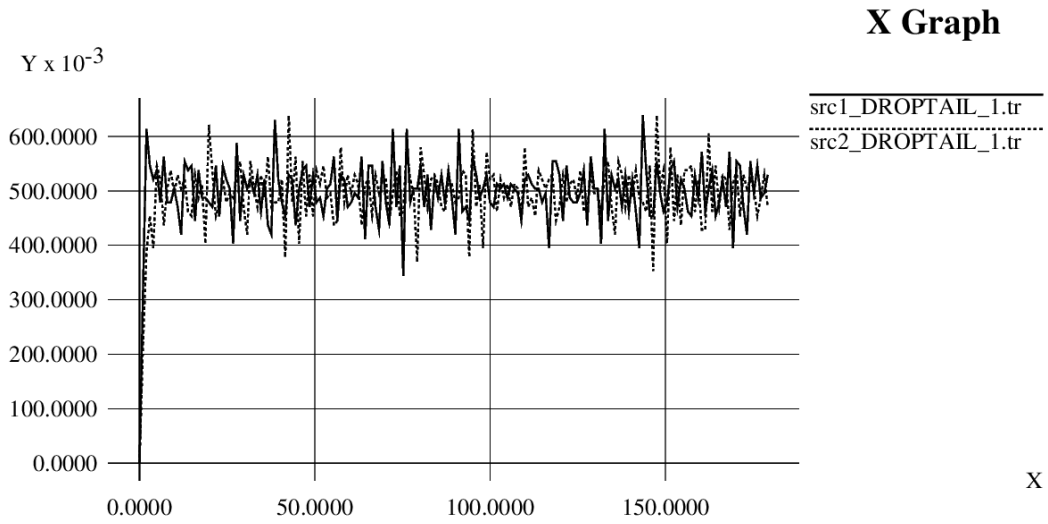
The node topology is as shown in the figure below. The link states are different, the link buffer management strategies are different, and the time interval is 1 second in order to calculate the instantaneous throughput.



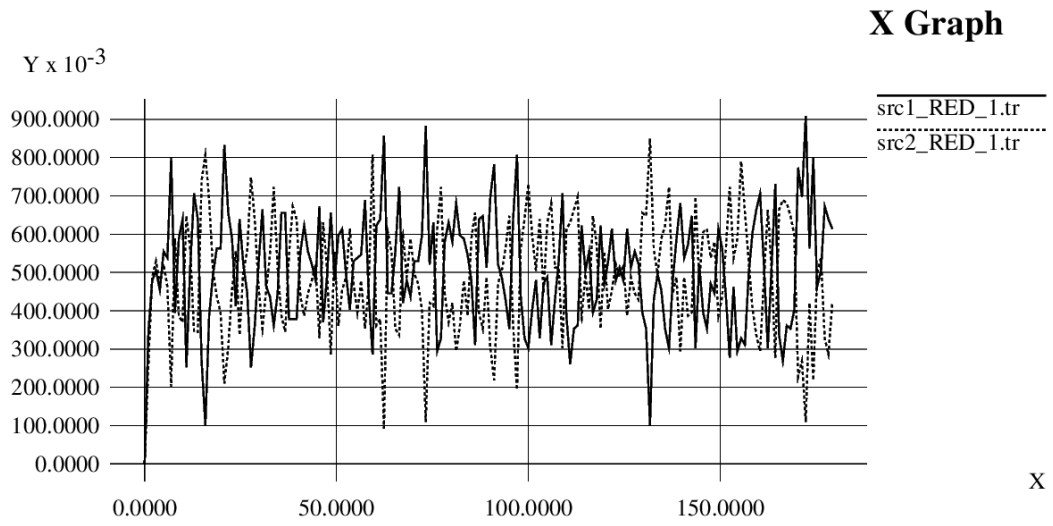
Average throughput (DROPTAIL, RED):

	Average Throughput (SOURCE 1) (Mbps)	Average Throughput (SOURCE 2) (Mbps)
DROPTAIL	0.4975 MBits/s	0.4943 Mbits/s
RED	0.5046 Mbits/s	0.4861 Mbits/s

a) DROPTAIL buffer management: Instantaneous throughput

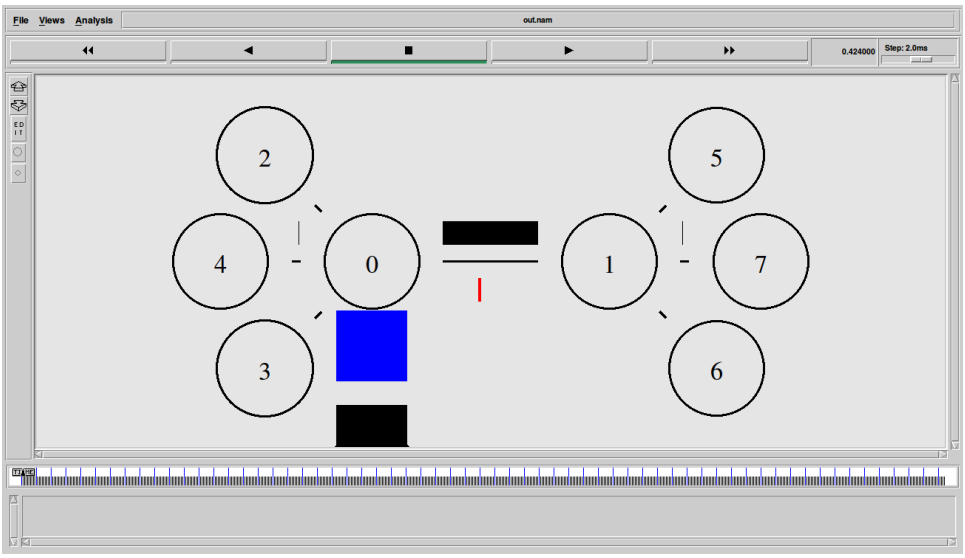


b) RED buffer management: Instantaneous throughput



Scenario 2:

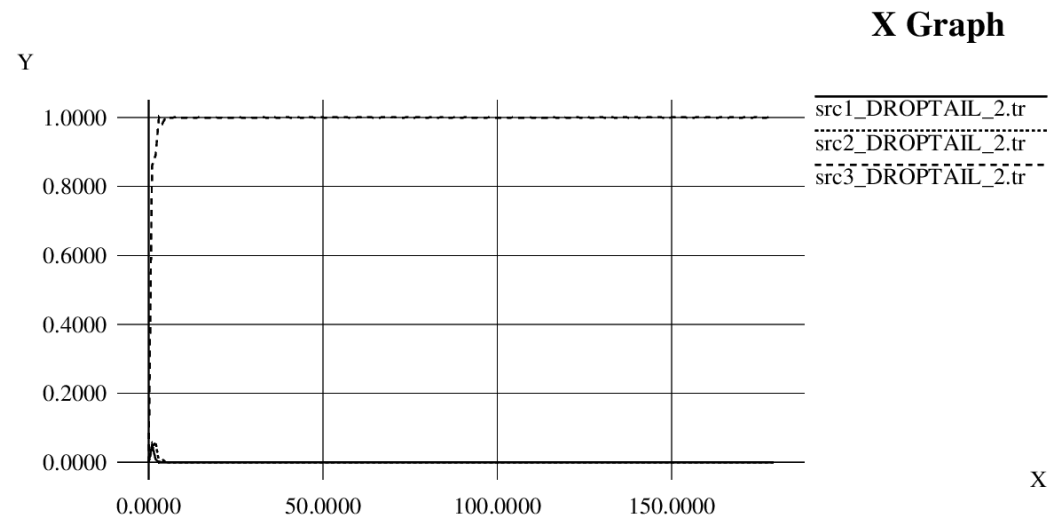
We add two other nodes in this case, and add UDP the source node. As for the throughput of this new node. The topology is as shown below



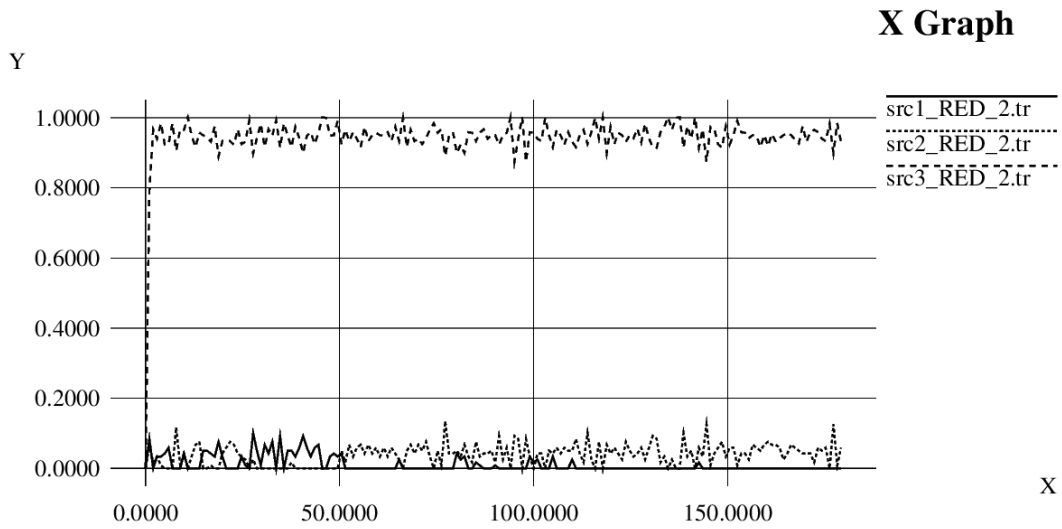
Average throughput (DROPTAIL, RED):

	Average Throughput (SOURCE 1) (Mbps)	Average Throughput (SOURCE 2) (Mbps)	Average Throughput (SOURCE 2) (Mbps)
DROPTAIL	0.0003 MBits/s	0.0006 Mbits/s	0.9934 Mbits/s
RED	0.0113 Mbits/s	0.0382 Mbits/s	0.9423 Mbits/s

a) DROPTAIL buffer management: Instantaneous throughput



b) RED buffer management : Instantaneous throughput



Final analysis:

Drop Tail is a Passive Queue Management (PQM) algorithm which only sets a maximum length for each queue at router. Routers decide when to drop packets. It uses first in first out algorithm. In Drop Tail, the traffic is not differentiated. Each packet has the same priority. When the queue buffer is filled to its maximum capacity, the packets arrived afterward are dropped till the queue is full. That is, Drop Tail will keep discarding/dropping the packet until the queue has enough room for new packets.

However, in RED, dropping is based on the threshold values; minimum threshold T(min) and maximum threshold T(max). RED monitors the average queue size avg, and checks whether it lies between some minimum threshold and maximum threshold. If it does, then arriving packet is dropped or marked with probability p=p(avg) which is an increasing function of the average queue size. If avg exceed T(max) , all the packet arrived will be dropped/discarded.

And for DropTail Scenario1, the router does not distinguish the two incoming links, and drop the packets when the queue is full. And as long as the packets are lost, the tcp agent will halve the congestion window, and in this way, two links will finally get the average throughputs near 500Kbits/second.

And for RED Scenario1, because the dropping probability of RED is calculated as follows: if count is the number of packets which are arriving consecutively and not discarded since the last discard packet. It has been sure that if count increases, then dropping probability will increase. So when one source node H1 transmits for a consecutively larger number of packets, then it has higher probability that the router will discard its packet. Thus, the result is that as long as this node's packet is discarded, H1 will halve it congestion window, and so the transmission rates goes down, and H2's transmission rates will goes up. And when H2's consecutively transmited number of packets goes up, it will be more probably that its packets will be dropped which result in the halving of the congestion window size. So in this way, H1 and H2's instantaneous throughputs will goes up and down more drastic than that of DropTail.

And for DropTail Scenario2, we introduce an additional UDP link here. Because UDP never react to the packets loss as TCP, thus, whenever there is packets loss, UDP just send the packets as before, however, TCP agents will halve their congestion windows. Because in this Scenario, the send rate of UDP is 1Mbits/second = the capacity of the link (R1,R2), so packets loss would always happen as long as any TCP agents send packets. As a result, TCP would decrease their window size to 0. And all the bandwidth is occupied by UDP. And so the throughput is H1=0, H2=0, H3=1000Kbits.

As for the RED Scenario2, although TCP agents will decrease their window sizes as in the above, the RED would drop the packets of UDP at a much higher probability. As a result, it is likely that sometimes, TCP agents will successfully send their packets and these packets will not be dropped. So the final result is that: TCP agents have much smaller throughputs. And UDP occupies almost all of the link bandwidth.

Appendix:

```
# Write ns2.tcl code for the specified network
# usage:
# ns ns2.tcl <QUEUE_MECHANISM> <SCENARIO_NO>
# example: ns ns2.tcl DROPTAIL 2
set ns [new Simulator]

set f0 [open src1_[lindex $argv 0]_[lindex $argv 1].tr w]
set f1 [open src2_[lindex $argv 0]_[lindex $argv 1].tr w]
set f2 [open src3_[lindex $argv 0]_[lindex $argv 1].tr w]
set nf [open out.nam w]
$ns namtrace-all $nf

set sum_1 0
set sum_2 0
set sum_3 0
set count 0
if {[lindex $argv 1] == 1} {
        proc finish {} {
                global f0 f1 argv sum_1 sum_2 count
                close $f0
                close $f1
                puts "Average throughput (src1) = [expr $sum_1/$count] MBits/s "
                puts "Average throughput (src2) = [expr $sum_2/$count] MBits/s "
                #exec nam out.nam &
                #Call xgraph to display the results
                exec xgraph src1_[lindex $argv 0]_[lindex $argv 1].tr src2_[lindex
$argv 0]_[lindex $argv 1].tr -geometry 800x400 &
                exit 0
        }
```

```
} elseif {[lindex $argv 1] == 2} {
        proc finish {} {
                global f0 f1 f2 argv sum_1 sum_2 sum_3 count
                close $f0
                close $f1
                close $f2
                puts "Average throughput (src1) = [expr $sum_1/$count] MBits/s "
                puts "Average throughput (src2) = [expr $sum_2/$count] MBits/s "
                puts "Average throughput (src3) = [expr $sum_3/$count] MBits/s "
                #exec nam out.nam &
                #Call xgraph to display the results
                exec xgraph src1_[lindex $argv 0]_[lindex $argv 1].tr src2_[lindex $argv 0]_[lindex $argv 1].tr src3_[lindex $argv 0]_[lindex $argv 1].tr -geometry 800x400 &
                exit 0
        }
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]


et queue "DropTail"
# Setting RED parameters
if {[lindex $argv 0] == "RED"} {
set queue "RED"
Queue/RED set thresh_ 10
Queue/RED set maxthresh_ 15
```

```
Queue/RED set linterm_ 50
}
$ns duplex-link $n0 $n2 10Mb 1ms $queue
$ns duplex-link $n1 $n2 10Mb 1ms $queue
$ns duplex-link $n3 $n2 1Mb 10ms $queue
$ns duplex-link $n4 $n3 10Mb 1ms $queue
$ns duplex-link $n5 $n3 10Mb 1ms $queue

# Queue limit for the link
$ns queue-limit $n2 $n3 20

if {[lindex $argv 1] == 2} {
        $ns duplex-link $n6 $n2 10Mb 1ms $queue
        $ns duplex-link $n7 $n3 10Mb 1ms $queue

        #Create a UDP agent and attach it to node n6
        set udp0 [new Agent/UDP]
        $ns attach-agent $n6 $udp0
        # Create a CBR traffic source and attach it to udp0
        set cbr0 [new Application/Traffic/CBR]
        $cbr0 set PacketSize_ 100
        $cbr0 set rate_ 1Mb
        $cbr0 attach-agent $udp0
        #set null0 [new Agent/Null]
        #$ns attach-agent $n7 $null0
        set sink2 [new Agent/LossMonitor]
        $ns attach-agent $n7 $sink2
        #$ns connect $udp0 $null0
        $ns connect $udp0 $sink2
}
#Create a TCP agent and attach it to node n0
set tcp0 [new Agent/TCP/Reno]
```

```
$ns attach-agent $n0 $tcp0
# Create a FTP traffic source and attach it to tcp0
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
#Create a TCP agent and attach it to node n1
set tcp1 [new Agent/TCP/Reno]
$ns attach-agent $n1 $tcp1
# Create a FTP traffic source and attach it to tcp1
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1

set sink0 [new Agent/TCPSink]
set sink1 [new Agent/TCPSink]
$ns attach-agent $n4 $sink0
$ns attach-agent $n5 $sink1
$ns connect $tcp0 $sink0
$ns connect $tcp1 $sink1

if {[lindex $argv 1] == 1} {
        proc record {} {
                global sink0 sink1 f0 f1 sum_1 sum_2 count
                #Get an instance of the simulator
                set ns [Simulator instance]
                #Set the time after which the procedure should be called
        again
                set time 0.99
                #How many bytes have been received by the traffic              sinks?
                set bw0 [$sink0 set bytes_]
                set bw1 [$sink1 set bytes_]
                #Get the current time
                set now [$ns now]
                #Calculate the bandwidth (in MBit/s) and write it to the
        files
```

```
                set sum_1 [expr $sum_1 + $bw0/$time*8/1000000];
                set sum_2 [expr $sum_2 + $bw1/$time*8/1000000];
                set count [expr $count + 1];
                puts $f0 "$now [expr $bw0/$time*8/1000000]"
                puts $f1 "$now [expr $bw1/$time*8/1000000]"
                #Reset the bytes_ values on the traffic sinks
                $sink0 set bytes_ 0
                $sink1 set bytes_ 0
                #Re-schedule the procedure
                $ns at [expr $now+$time] "record"
        }
} elseif {[lindex $argv 1] == 2} {
        proc record {} {
                global sink0 sink1 sink2 f0 f1 f2 sum_1 sum_2 sum_3 count
                #Get an instance of the simulator
                set ns [Simulator instance]
                #Set the time after which the procedure should be called
        again
                set time 0.99
                #How many bytes have been received by the traffic          sinks?
                set bw0 [$sink0 set bytes_]
                set bw1 [$sink1 set bytes_]
                set bw2 [$sink2 set bytes_]
                #Get the current time
                set now [$ns now]
                #Calculate the bandwidth (in MBit/s) and write it to the
        files
                set sum_1 [expr $sum_1 + $bw0/$time*8/1000000];
                set sum_2 [expr $sum_2 + $bw1/$time*8/1000000];
                set sum_3 [expr $sum_3 + $bw2/$time*8/1000000];
                set count [expr $count + 1];
                puts $f0 "$now [expr $bw0/$time*8/1000000]"
                puts $f1 "$now [expr $bw1/$time*8/1000000]"
```

```
                puts $f2 "$now [expr $bw2/$time*8/1000000]"
                #Reset the bytes_ values on the traffic sinks
                $sink0 set bytes_ 0
                $sink1 set bytes_ 0
                $sink2 set bytes_ 0
                #Re-schedule the procedure
                $ns at [expr $now+$time] "record"
        }
}
$ns at 30.0 "record"
$ns at 0.0 "$ftp0 start"
$ns at 0.0 "$ftp1 start"
if {[lindex $argv 1] == 2} {
        $ns at 0.0 "$cbr0 start"
        $ns at 180.0 "$cbr0 stop"
        $udp0 set class_ 3
        $ns color 3 Green
}
$ns at 180.0 "$ftp0 stop"
$ns at 180.0 "$ftp1 stop"
$tcp0 set class_ 1
$tcp1 set class_ 2
$ns color 1 Blue
$ns color 2 Red
$ns at 180.0 "finish"
$ns run
```